2

Advanced Nulling Techniques

Final Report

Contract Number: F04701-86-C-0098

September 1, 1988

Atlantic Aerospace Electronics Corporation

6404 Ivy Lane
Suite 300
Greenbelt, MD 20770-1406
301-982-5200

470 Totten Pond Road
Waltham, MA
02154-1905
617-890-4200

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS ~~DOD Directive 5230.25~~
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT ~~DTIC Alexandria, VA 22314~~
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-1028	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Atlantic Aerospace Electronics Corporation 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION DCASMA, Baltimore
6c. ADDRESS *(City, State and ZIP Code)* 470 Totten Pond Road Waltham, MA 02154	7b. ADDRESS *(City, State and ZIP Code)* 300 East Joppa Road Hampton Plaza Bldg - Room 200 Towson MD 21204
8a. NAME OF FUNDING/SPONSORING ORGANIZATION USAF / AFSC HQ Space Division 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F04701-86-C-0098
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*
Advanced Nulling Techniques Final Report Phase 2

12. PERSONAL AUTHOR(S)
Dr. Michael S. Wengrovitz

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
Final Report	FROM 9/86 TO 9/88	1988, September 1	286

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Jammers, Environmental filters, Electronic Warfare open-loop processor, closed-loop processor, nulling, nulling weights, Signal extraction, Communications (cnr)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Project Summary

The objective of this work was to identify and demonstrate nulling techniques which are rapidly adapting and robust with respect to various types of errors. The principal application of these nulling techniques is to the communications problem which is characterized by frequency-hopped and other broadband signals, and a diversity of non-stationary jammers.

This report describes the work performed in support of developing advanced nulling techniques that are effective in non-stationary broadband environments. Areas of emphasis include:

1. Development of a class of open-loop nulling algorithms that converge much more rapidly than conventional algorithms in non-stationary environments with user signals present.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER *(Include Area Code)*	22c. OFFICE SYMBOL

19.

2. Unification and extension of this class of algorithms to multi-dimensional and broadband arrays.

3. Comparison of multiple-null and other techniques with alternative broadband methods including spatial/temporal filtering.

4. Assessment of the performance of these techniques in the face of various errors including user direction uncertainties and hardware errors.

5. Identification of an example high-level nulling processor.

Main results of this work include the development of a new class of nulling algorithms which are applicable to a class of arrays designated as convolutional arrays. These algorithms greatly improve transient response characteristics when a user signal is present in the sensor data and can be structured so that multiple-null and related techniques can be incorporated. The robustness of the new class of algorithms to mismatch caused by user direction and hardware errors is far superior to conventional nulling algorithms. Other main results include comparison of multiple-null techniques with other broadband adaptive nulling techniques, identification of methods which automatically place multiple nulls in more general spatial/temporal adaptive arrays, and high-level design of an example nulling system architecture.

Potential applications of the rapidly convergent algorithms are to communications systems which function in highly-dynamic broadband environments. Additional applications of the multiple-null techniques include adaptive radar and sonar systems.

Contents

Chapter 1

Executive Summary

Contents

List of Figures

1.1 Introduction

Conventional nulling techniques have been traditionally developed assuming a stationary interference environment. However, the modern "electronic battlefield" is anything but stationary. Rather it is characterized by frequency-hopped and other broadband communications signals, and a diversity of non-stationary jammers. The thrust of the reported research has been to identify nulling techniques that are fast and effective in non-stationary broadband environments.

The objective of this research was to identify algorithms applicable to the communications problem that would be both fast, and also robust despite inevitable hardware and other errors. The communications focus has exerted a strong influence on the work. The distinguishing feature of the communications problem is the presence of both user signals and interference signals in the data used to calculate the adaptive nulling weights. Thus, the nulling system must process data containing two types of signals so as to produce weights for an adaptive filter that will pass one class of signals with little or no distortion while strongly suppressing the other class.

To solve this problem, there must be some distinguishing characteristics of the desired signals. The characteristic that we have assumed is coarse direction information for the user signals. Such information might be available from user tracking, or else from a-priori information.

An attractive approach for designing a nulling system applicable to non-stationary environments is to employ a "two-tier" architecture. The first tier utilizes a fast open-loop nulling algorithm; the second tier utilizes a slower closed-loop algorithm which is capable of improved performance in stable environments. The overall system monitors the electromagnetic environment and upon detecting a significant change, activates the open-loop processor. This processor rapidly calculates a useful (albeit sub-optimum) set of nulling weights. These nulling weights then are used as initial conditions in the follow-on closed-loop processor. The work reported here has focused primarily upon the open-loop component of such a system. The reason for this focus has been twofold. First, open-loop processors by themselves represent effective solutions to many nulling problems. Second, the problem of designing effective open-loop processors for non-stationary environments is not nearly so well understood as that of designing closed-loop processors for slowly changing environments.

For purposes of the study, it was assumed that digital processing techniques were to be employed to calculate the nulling weights for the adaptive antenna. This seemed to be an appropriate assumption given the many recent advances in the areas of A/D converters, solid-state processors and parallel-processing architectures. Thus, each set of nulling weights was calculated using a sequence of data "snapshots" of the array sensor outputs. Each snapshot consisted of a set of complex amplitudes, each corresponding to the output of a different array sensor at a common instant.

1.2 Relationship to Prior Work

In previous work [1] we identified a class of open-loop nulling algorithms that converge much more rapidly than conventional open-loop algorithms (e.g. SMI) in non-stationary environments with user signals present. We also identified a second algorithm that automatically places multiple nulls on jammers rather than the simple nulls characteristic of conventional nulling techniques. The multiple nulls have the interesting property of providing jammer suppression over a frequency band much wider than the actual processing band. Both algorithms were developed for uniform linear arrays.

The thrust of the present work has been as follows:

1. Improve and unify the previously developed algorithms, and to extend the algorithms to more general (convolutional) arrays, and to two-dimensional arrays.
2. Compare the multiple-null and related techniques to alternative broadband techniques such as spatial/temporal filtering.
3. Assess the performance of the techniques given real-world effects such as errors in the assumed directions of the users and hardware errors.
4. Identify a suitable high-level architecture for a nulling processor.

1.3 Main Results

Our main results in the four areas are described in the following subsections.

1.3.1 Fast Open-Loop Algorithms

The main results with regard to extending the fast open-loop algorithms are:

1. Demonstrat‿n that the transient response of conventional nulling algorithms is slowed dramatically when a user signal is introduced into the data used to derive the nulling weights. For example, Figure 1.1 depicts the degradation in the transient response of the Generalized Sidelobe Canceller (GSC) algorithm [2][3] when a user signal is introduced in an example scenario consisting of a 16-element linear array operating in the presence of two jammers.

2. Recognition that the poor transient performance of many nulling algorithms is due to the fact that the algorithms tend to treat the user as a mainbeam jammer, and to null the user when a sample covariance matrix $\hat{R}$ is used in place of the asymptotic covariance matrix R. For example, Figures 1.2 and 1.3 contrast the GSC antenna patterns after 100 snapshots for the foregoing scenario with user absent and user present.

3. Development of a new class of nulling algorithms applicable to convolutional arrays, which exhibits a greatly improved transient response when a user signal is present in the sensor data. The essence of the algorithms is to exploit a-priori information on the user directions to spatially pre-filter the user signal out of the array snapshots; the nulling weights then are calculated from the residual data (filtered snapshots). For example, Curve 3 of Figure 1.4 depicts the transient response of the new type of algorithm for a scenario involving a (two-dimensional) 6×6 planar array. Note that the transient response of the new algorithm with user present is better than that of conventional Sample-Matrix-Inversion (SMI) technique[4] with no user.

4. Structuring of the new class of algorithms so that multiple-null and related techniques can be incorporated. Specifically these techniques can be applied by operating either on snapshot data or on the estimated covariance matrix.

1.3.2 Broadband Performance

Two types of adaptive arrays for broadband applications were considered: 1) arrays with weight-and-sum processors, and 2) arrays with filter-and-sum processors. An array with a weight-and-sum processor is designated as a *narrowband array* since this type of array typically has been used in narrowband applications. An array with a filter-and-sum processor is designated as a *broadband array*. The two types of array are illustrated in Figures 1.5.

Both types of arrays are useful in broadband applications. The weight-and-sum pro-

Figure 1.1: Performance of the Generalized Sidelobe Canceller (GSC) With and Without User Present

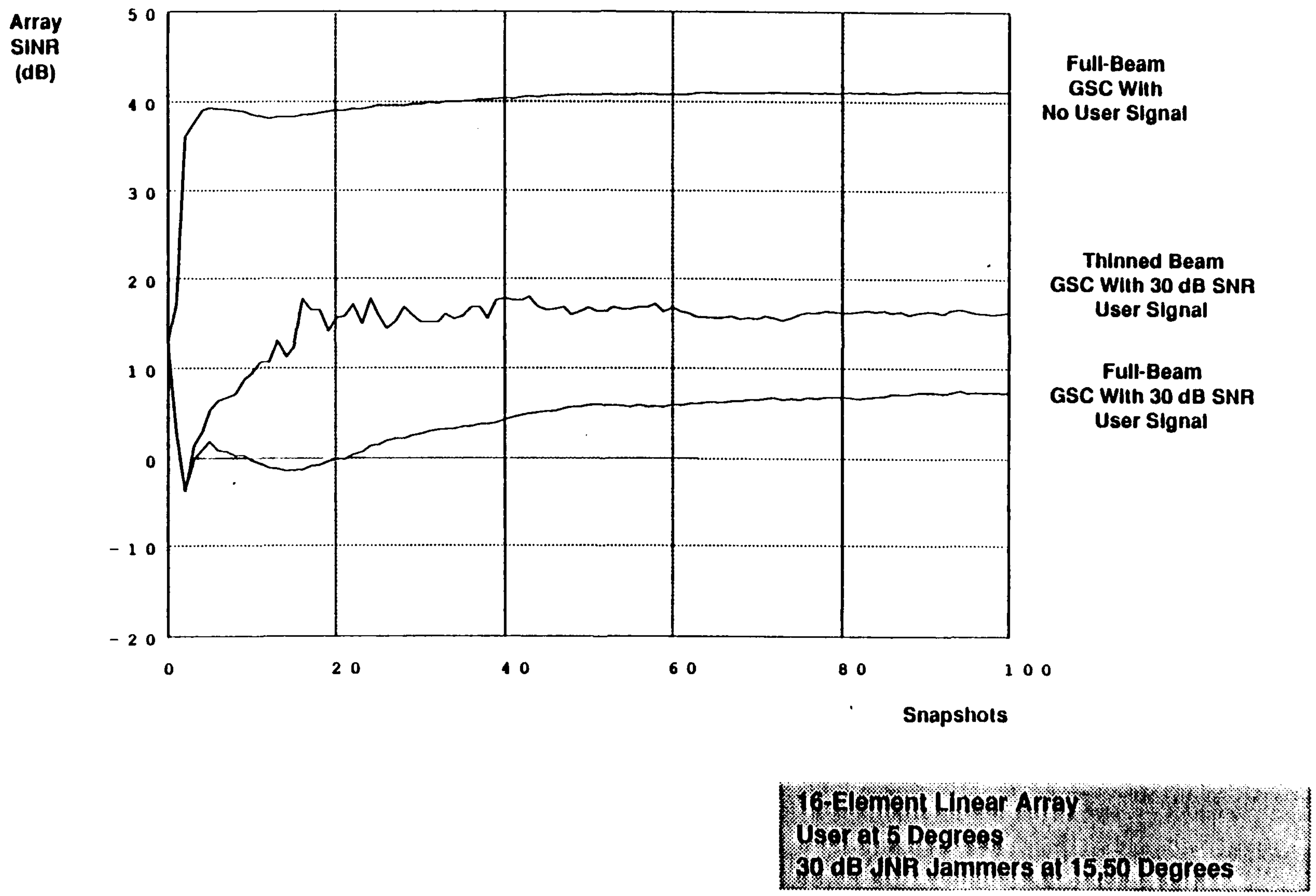


Figure 1.2: Antenna Pattern After 100 Snapshots Without User Present

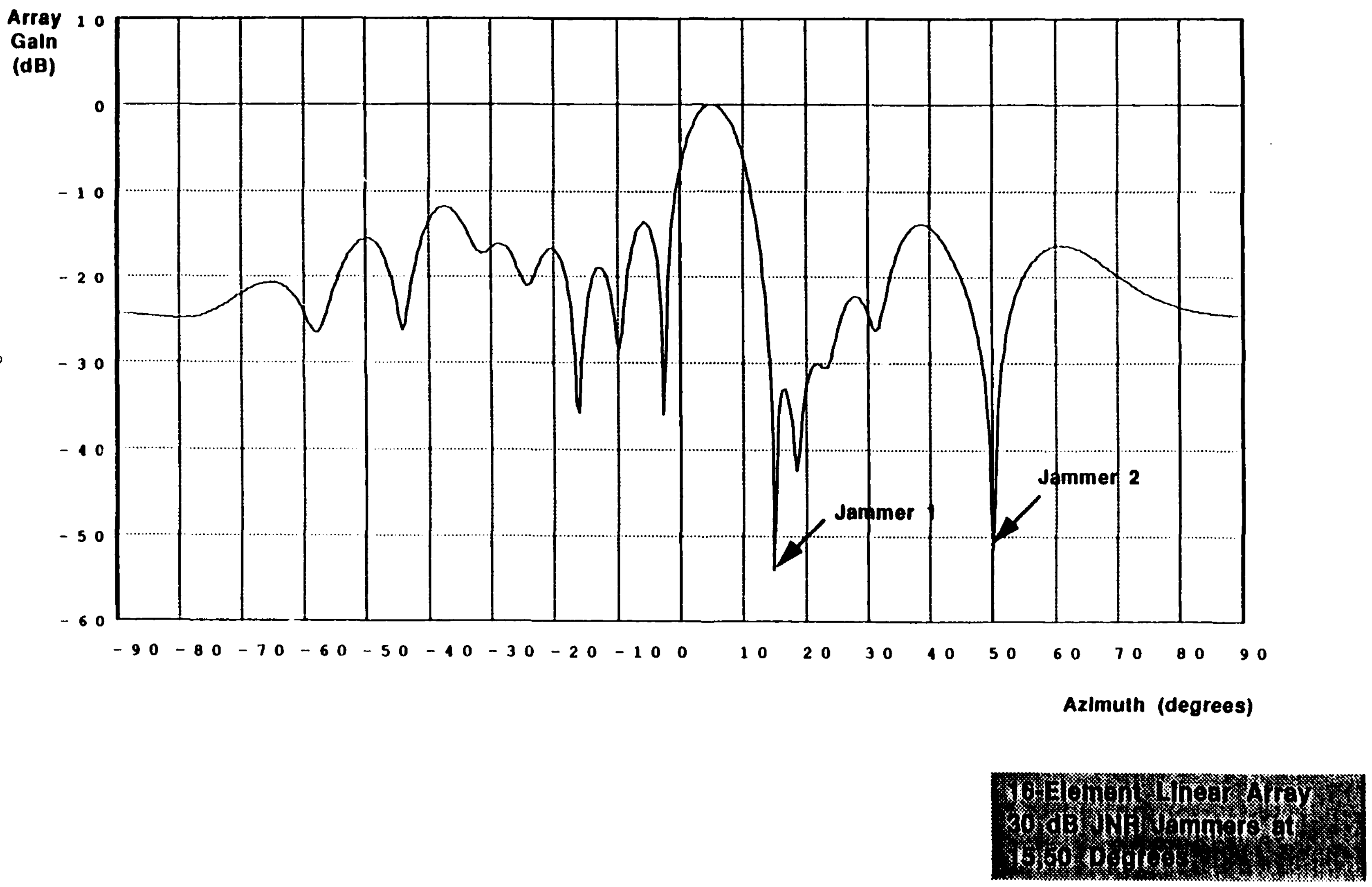


Figure 1.3: Antenna Pattern After 100 Snapshots With User Present

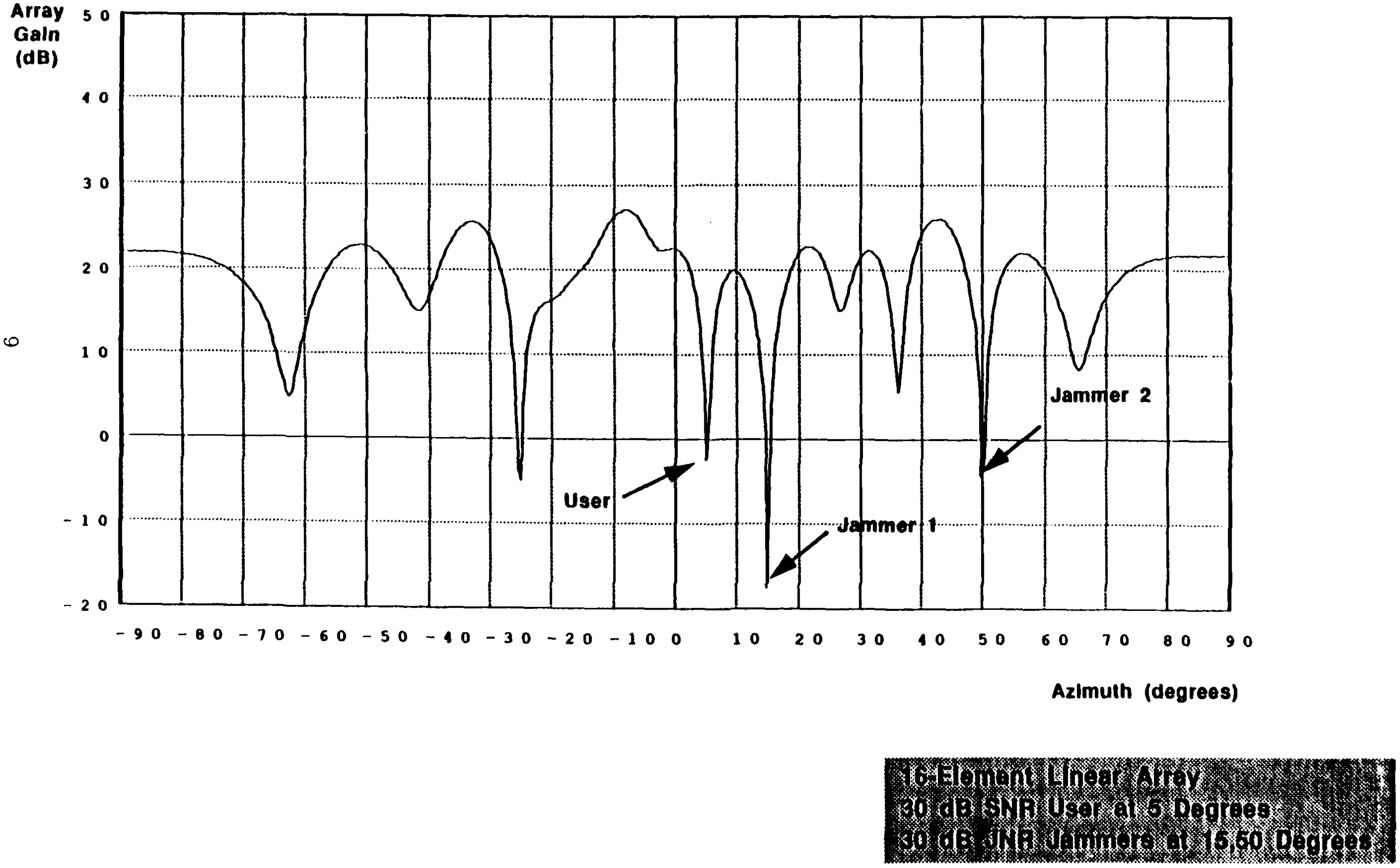
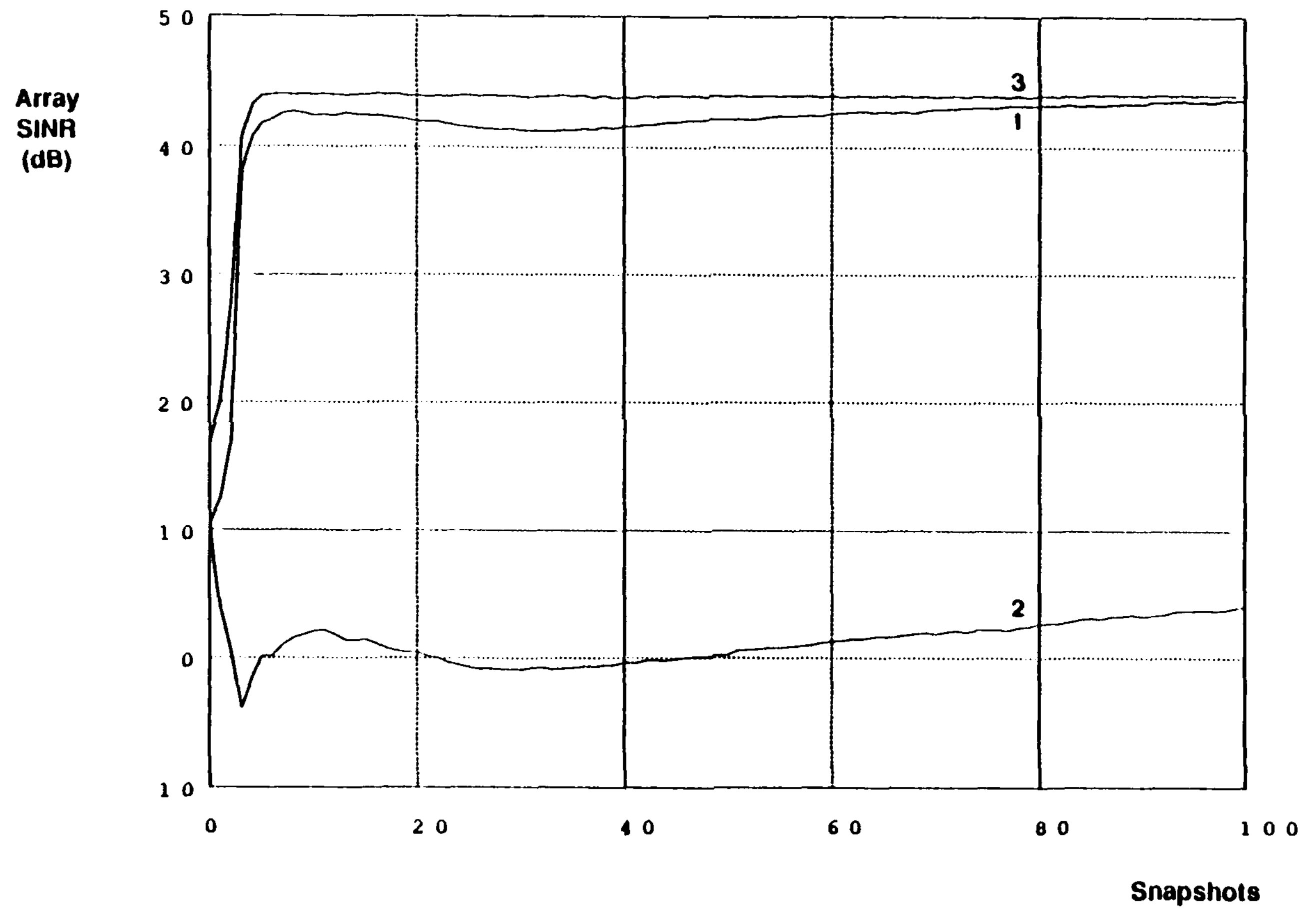


Figure 1.4: Example of Improvement in Planar Array Transient Response with New Algorithm



Curve 1: SMI Without User Present
Curve 2: SMI With 30 dB SNR User Present
Curve 3: New Algorithm With 30 dB SNR User Present

6x6 Planar Array
30 dB JNR Jammers
at Az/El's of: 0/60, 135/45 , 270/33.5

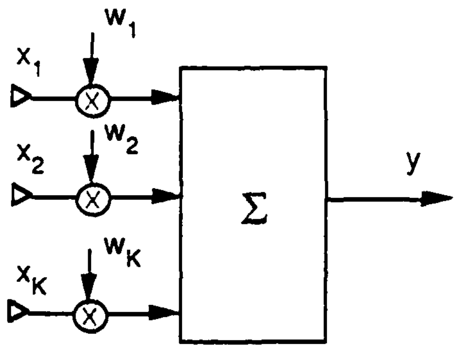


Figure 1.5a: Narrowband Array (Weight-and-Sum Processor)

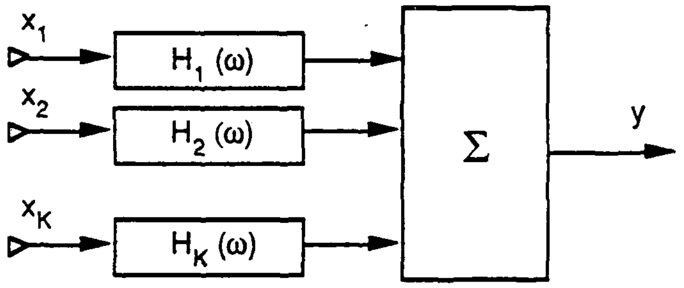


Figure 1.5b: Broadband Array (Filter-and-Sum Processor)

cessor multiplies each sensor output by a constant *frequency-independent* complex-valued weight prior to summation. By contrast, the filter-and-sum processor in a broadband array multiplies each sensor output by a *frequency-dependent* complex-valued weight. The frequency-dependent weights each can be realized as a bank of narrowband filters; alternately the frequency-dependent weights can be realized via transversal filters.

For a fixed number of sensors, a broadband array has a larger number degrees of freedom due to the availability of of samples in both space and time. Therefore, a broadband array has greater adaptive potential. On the other hand, narrowband arrays potentially are of interest in applications in which the number of sensors exceeds the number of jammers so that "extra" degrees of freedom already are available without the introduction of frequency-dependent weights.

Since transient response issues are treated separately (see Sections 1.3.1 and 1.3.3), we primarily considered here the comparative steady-state performance of narrowband arrays and broadband arrays in broadband settings.

Our major conclusions relating to broadband performance are as follows:

1. Multiple linear constraints are required in conjunction with power-minimization algorithms to determine effective nulling weights. The role of the constraints is to ensure constant gain and group delay across the frequency band of interest (use of a gain constraint at a single frequency only causes the user signal to be strongly attenuated at other frequencies). In effect the processor treats signal components at other frequencies as mainbeam jammers, and selects weights that attenuate these frequencies. The result is severe distortion of the user signal. Figure 1.6 illustrates the constant gain and group delay versus frequency that is achievable using suitably selected linear constraints in a broadband array.

2. Multiple-null techniques, or related methods for producing clustered nulls, potentially are useful with regard to providing enhanced broadband performance with narrowband arrays. Figure 1.7 illustrates clustered nulls for tone jammers. Note that clustered nulls extend the angular sector of jammer suppression well beyond that of the angles at which the jammers are located. Figures 1.8 and 1.9 illustrate the character of multiple nulls produced in two-dimensional arrays. Note that suppression is provided in extended angular sectors centered upon the actual jammer locations.

3. The multiple null techniques developed in Phase I of our work[1] typically produce

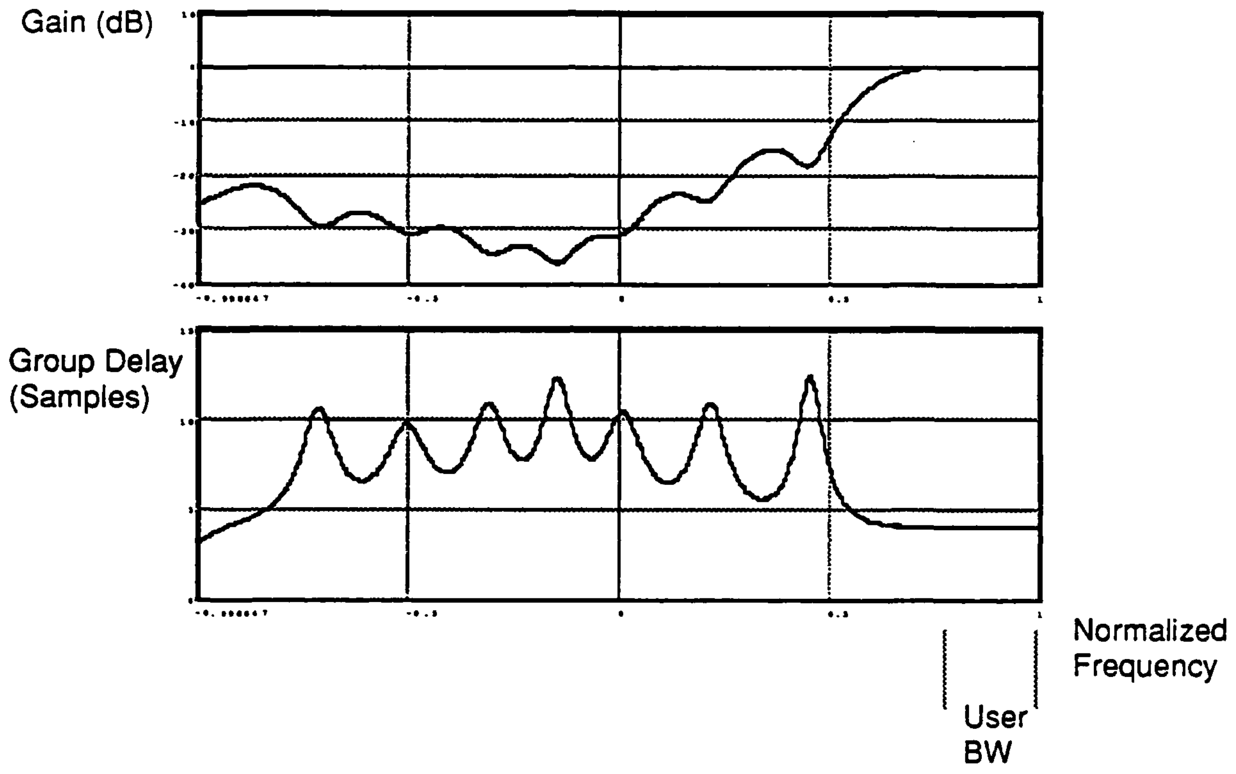


Figure 1.6: Frequency Response of a Linear Broadband Array in the User Look Direction.

16 Element broadband array with 8 taps per element. 4 constraints were used to steer the array toward a 25% fractional bandwidth user at 10 degrees.

Figure 1.7: Beampattern for a Linear Array With Clustered Null Processing

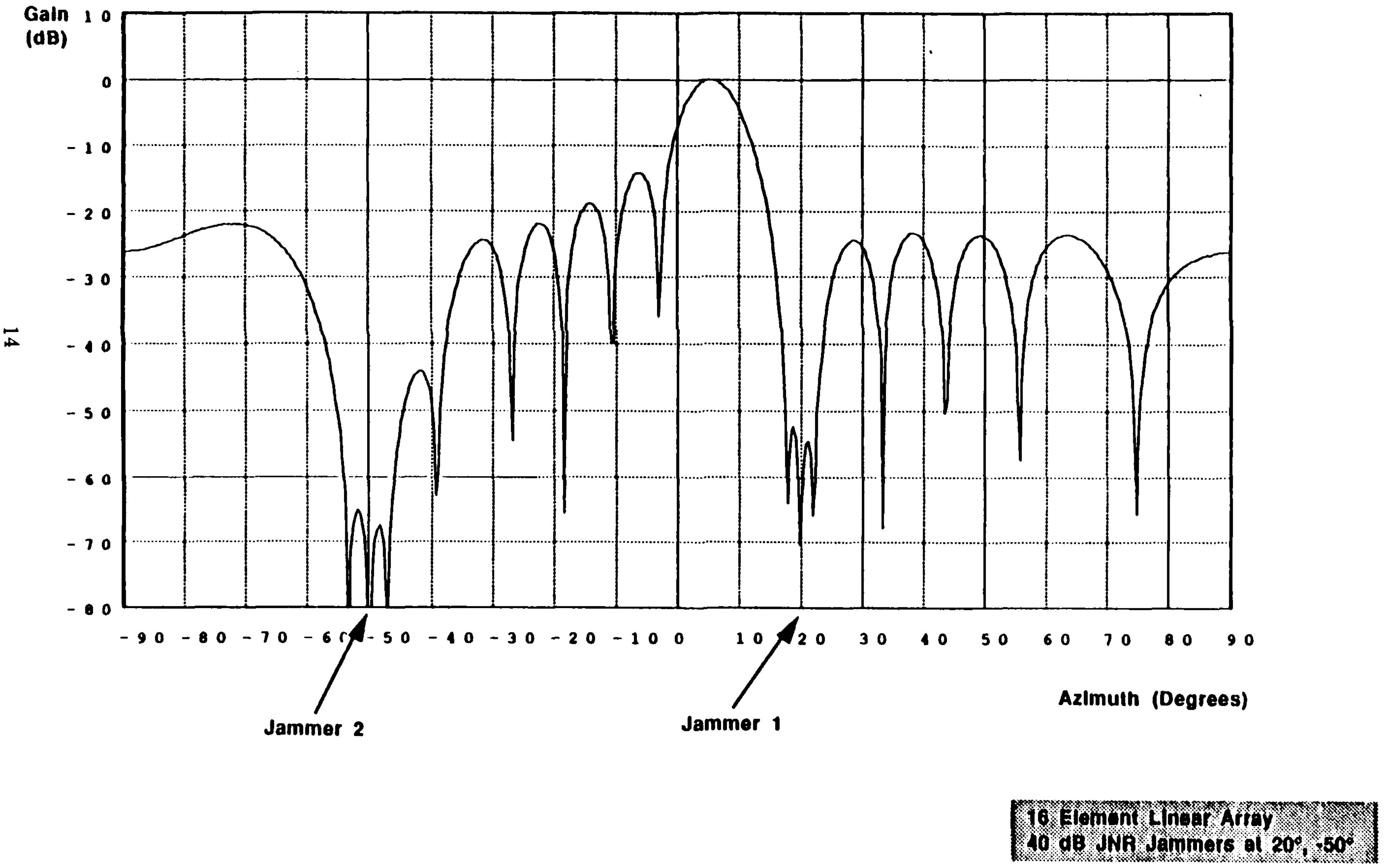


Figure 1.8: Beampattern & Derivative (at Elevation = 60 Degrees) for a Planar Array With Higher-Order Null Processing

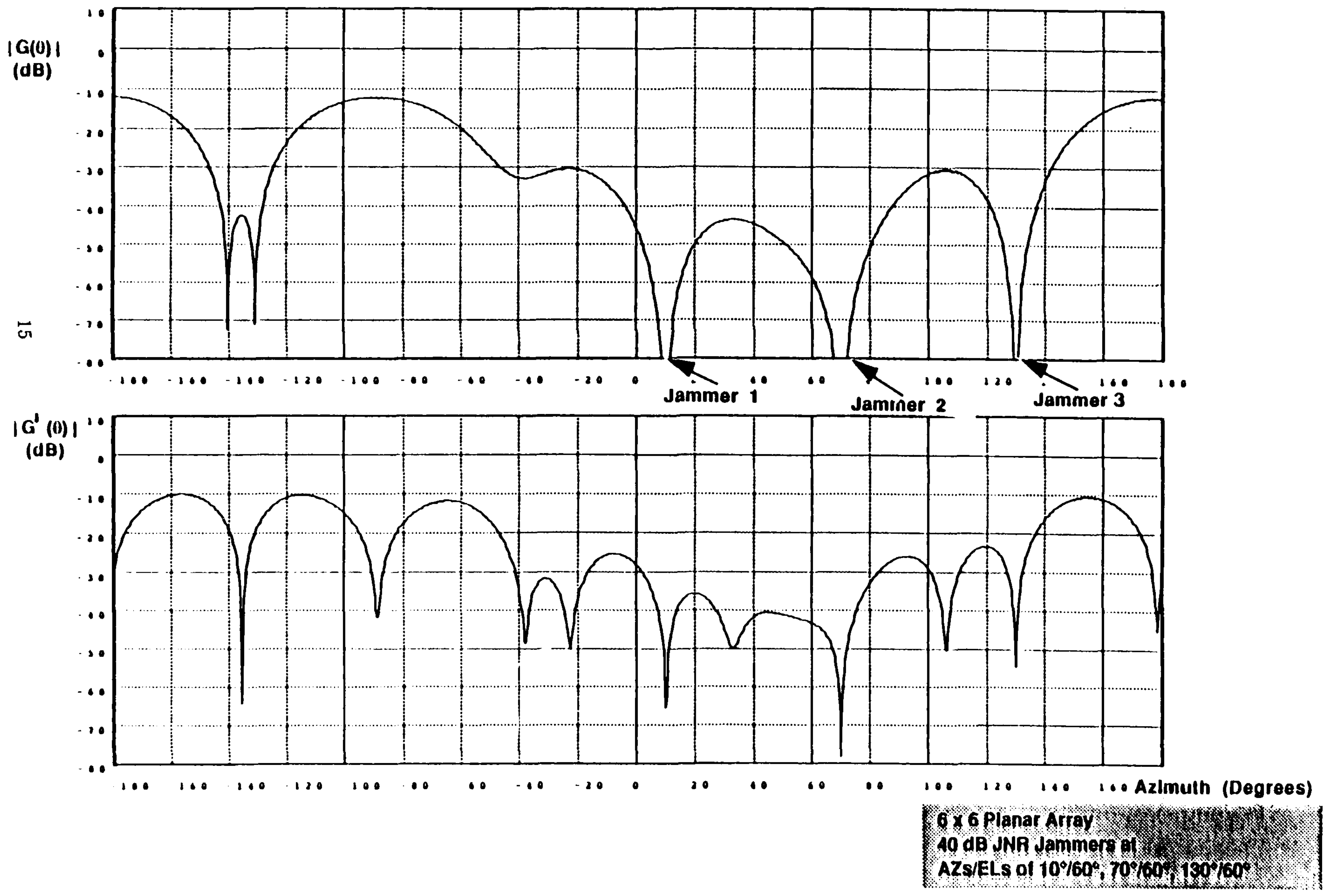
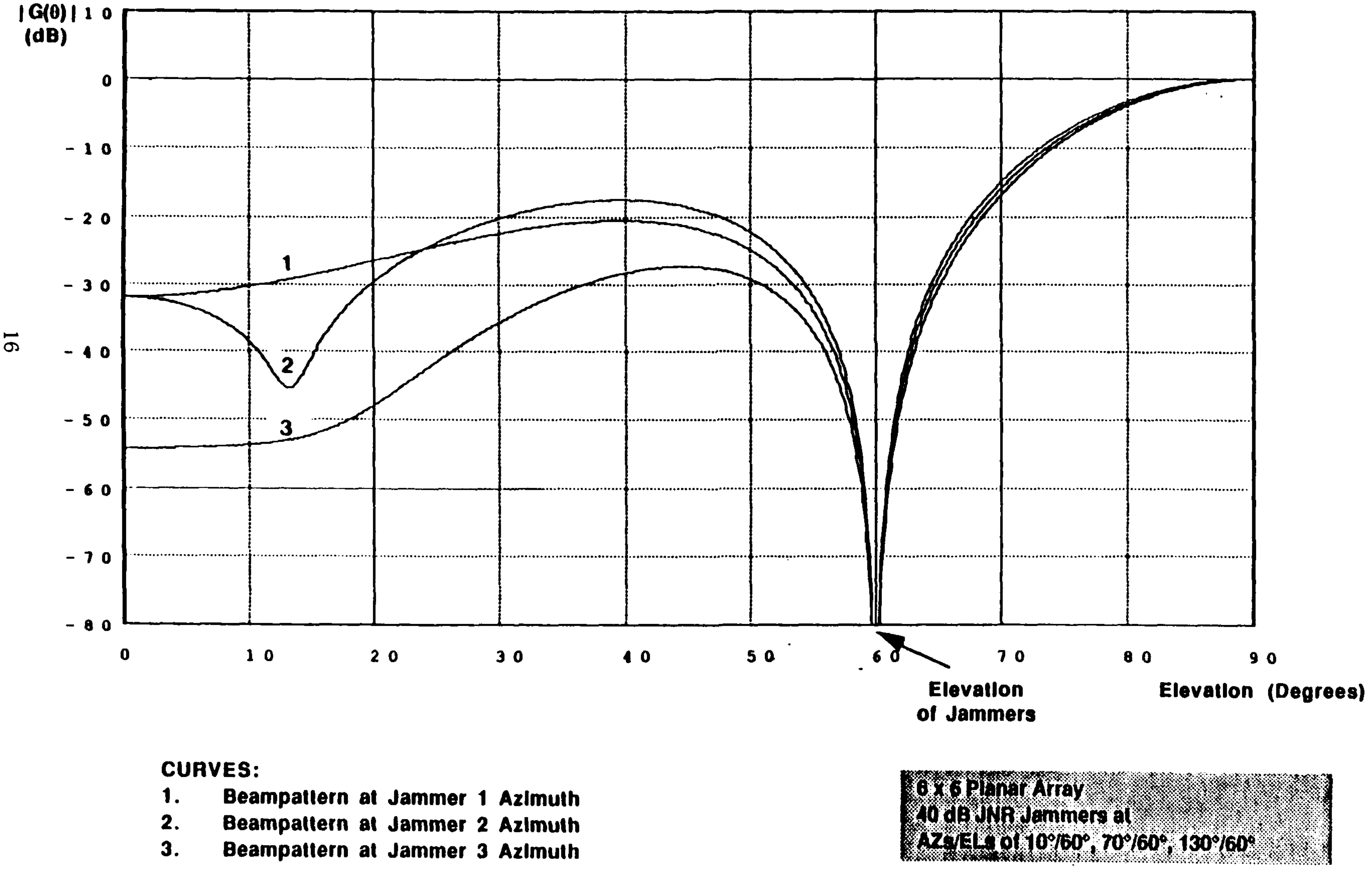


Figure 1.9: Beampattern (at Jammer Azimuths) for a Planar Array With Higher-Order Null Processing



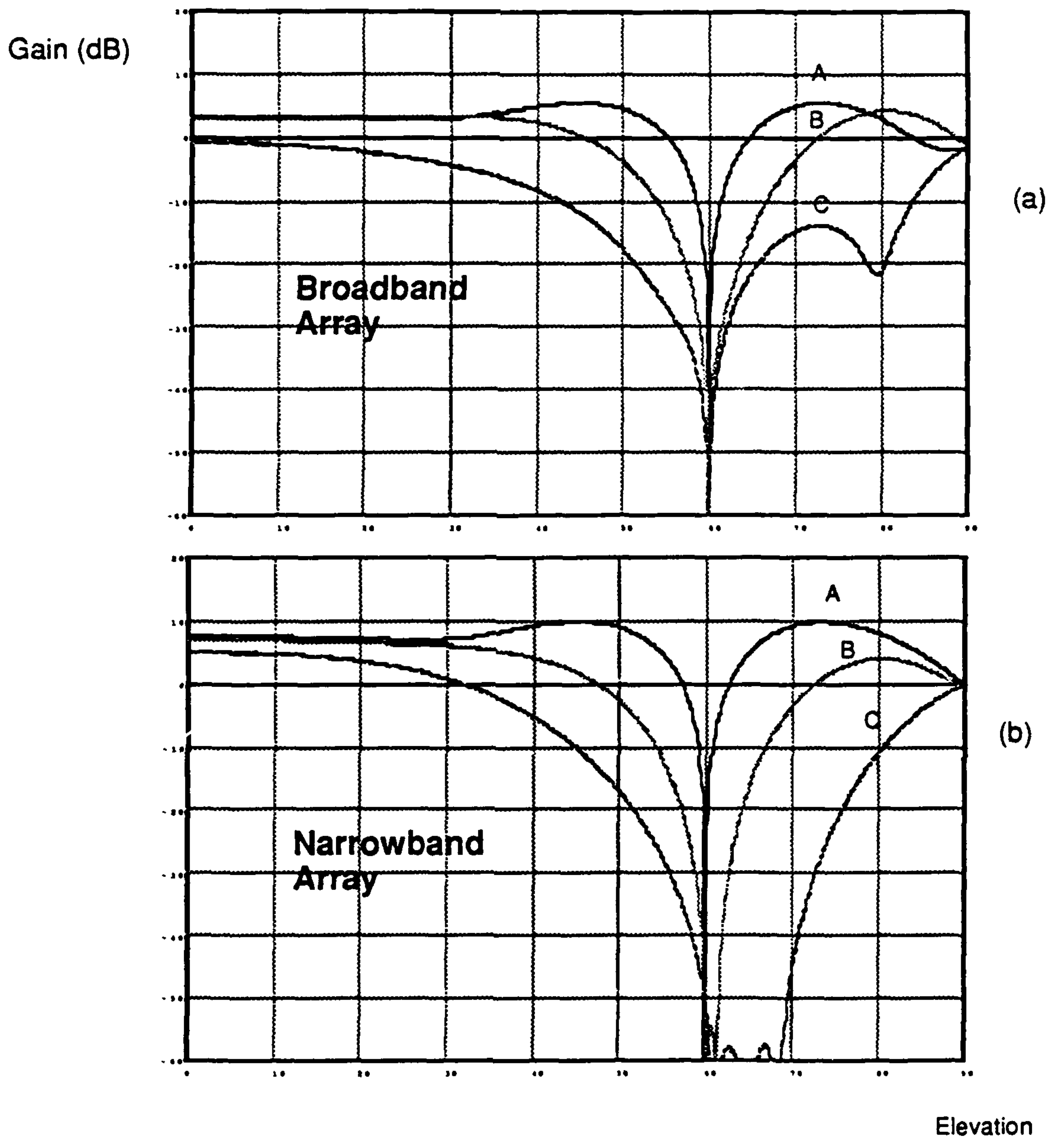
nulls both in azimuth and elevation for two-dimensional arrays. However, only the elevation nulls are useful in improving broadband performance; the azimuth nulls simply consume additional degrees of freedom. A related technique was identified in Phase II that produces extended nulls in the elevation coordinate only, and produce enhanced broadband performance with only one-half the number of degrees of freedom of multiple nulls. However, computation of the weights required to produce the nulls is more involved than that for multiple nulls.

4. Multiple nulls potentially are useful in avoiding the nulling system problems that result from the presence of a user signal. Specifically, the potential exists to filter the user signal out of the sensor data using band-pass filters, and to then determine the nulling weights based upon the jamming signals in the adjacent frequency bands. Since multiple nulls provide spectral extension of nulls, the resultant weights also suppress the in-band jamming signals. Multiple nulls also potentially are useful in problems in which the array is mounted on a platform that can significantly change its angular orientation over the time interval required to calculate a new set of nulling weights.

5. Due to the sectoral suppression of jammers, multiple-null and elevation-null techniques are not well-suited for suppressing a jammer close to the user since the techniques tend to suppress not only the jammer, but also the user.

6. In contrast to narrowband arrays, broadband arrays produce sharp spatial nulls on jammers; that is, the nulls are of very limited extent both in azimuth and in elevation. Thus, broadband arrays are better suited than narrowband arrays for suppressing broadband jammers that are located close to the user. For example, Figure 1.10 contrasts the elevation beam patterns of a narrowband and a broadband array for a 6×6 two-dimensional arrangement of sensors. The three jammers have fractional bandwidths of 0% (tone jammer), 5% and 30% respectively. Note that whereas the narrowband array must produce extended suppression in elevation to effectively suppress the broadband jammers, the broadband array is able to do so with point suppression in elevation.

7. Techniques also have been identified for producing multiple nulls with broadband arrays. Multiple nulls in broadband arrays potentially are useful in the same ways as for narrowband arrays; that is, to avoid problems associated with presence of the user signal, and also to reduce the impact of platform motion.



6x6 2D narrowband array with omni-directional quiescent pattern. Azimuth/Elevation of the 3 jammers was 30/60(A) ,140/60(B), and 280/60(C). Fractional bandwidth of jammers was 0%(A) ,5% (B) ,30%(C). All jammers were 0 dB. 40 dB JNR.

Figure 1.10

Beampatterns (at f=1.0) at Jammer Azimuths for Optimal Planar Broadband and Narrowband Arrays.

8. Owing to the large number of degrees of freedom, broadband arrays can require substantial computation to determine nulling weights. Thus, it is highly desirable to employ beamformers with thinned beam sets to reduce the required computation.

1.3.3 Sensitivity to Errors

The main results for pointing and hardware errors are as follows.

1. The presence of a user signal in the data used to calculate nulling weights greatly magnifies the adverse impacts of pointing errors and hardware errors. For example, Figure 1.11 depicts the steady-state response of an adaptive array, using an optimum Wiener processor in a scenario in which the processor assumes the user to be located at an off-broadside angle of $\theta = 5°$, and the user is actually at an angle of $\theta = \theta_U$, creating a pointing error (or mismatch) of $\theta_U - 5°$. Curves 1 - 4 depict the relative gains of the array in the user direction as the mismatch $\theta_U - 5°$ is varied, for four levels of the user signal. Clearly the sensitivity to mismatch increases with the level of the user signal, and becomes extreme for a user SNR of 30 dB.

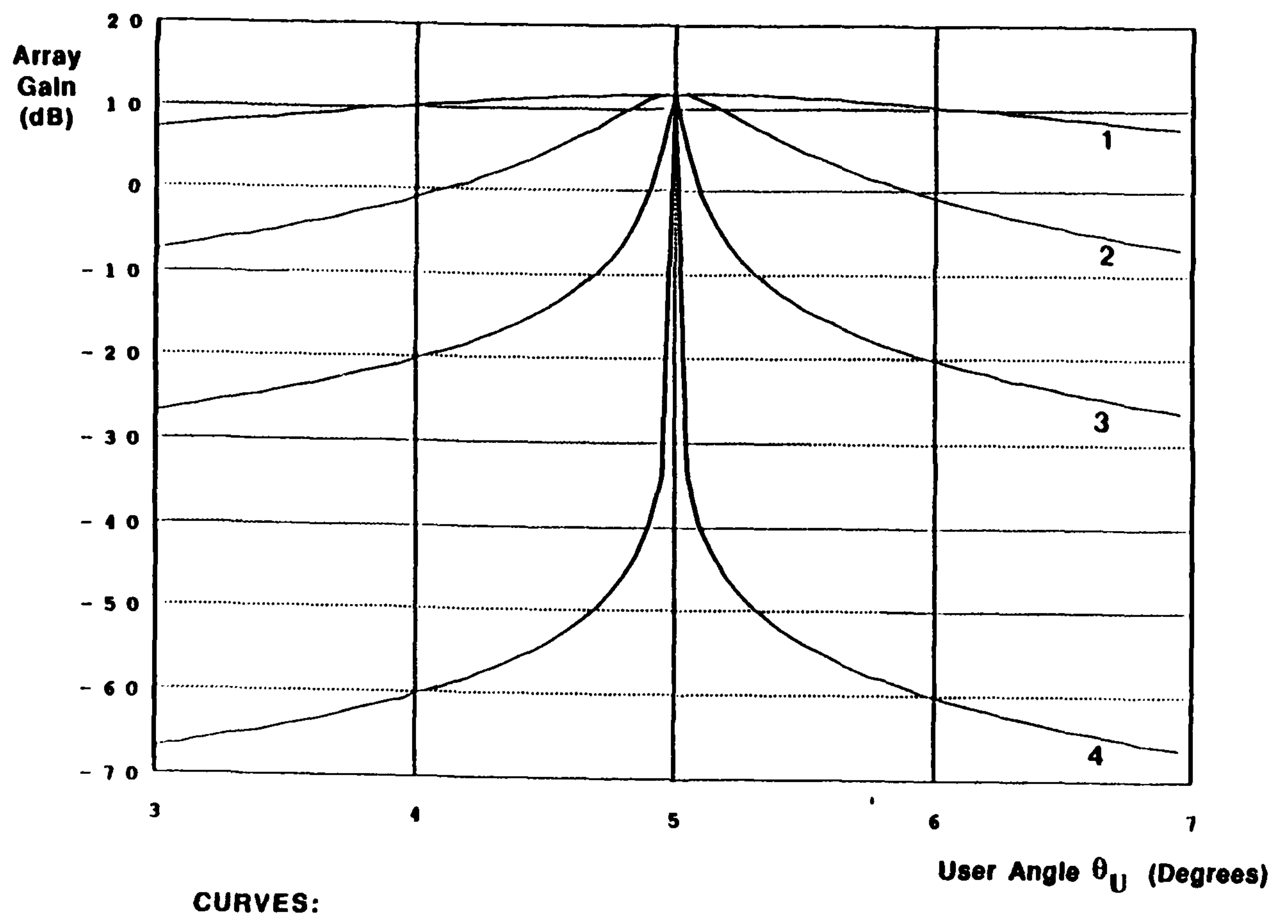
2. The effect of such errors is to change the apparent angle of arrival of the user signal; the nulling processor then mistakes the user signal for a mainbeam jammer and attempts to ***null the user along with the jammers.*** For example, Figures 1.12a and 1.12b depict the antenna patterns in the foregoing scenario for a user SNR = 30 dB and a mismatch of only 0.1 degree, or approximately 1/120 beamwidth.

3. The steady-state interference-suppression performance of the new class of algorithms described in Chapter 3 is much less sensitive to both pointing errors and channel hardware errors than many conventional algorithms in the user-present case. This property is a consequence of the fact that the new class of algorithms filters the user signal(s) out of the data prior to calculating nulling weights.

   For example, for the foregoing scenario Figure 1.13 compares the user mismatch sensitivity of versions of the new algorithms (Curves 3 and 4) to that of the Wiener processor (Curves 1 and 2) for the case of a user with SNR = 30 dB. It is evident that the new algorithms greatly reduce sensitivity.

4. A variety of "fixes" have been proposed in the literature to reduce the sensitivity of conventional null algorithms to the foregoing types of errors[5][6][7][8][9][10]. Under steady-state conditions the performance of the new class of algorithms with no fixes

Figure 1.11: Dependence of Pointing Error Sensitivity on User Signal Level



CURVES:

1. User Signal Absent
2. SNR = -10 dB
3. SNR = 0 dB
4. SNR = 30 dB

16-Element Linear Array
Variable SNR User Signal Present at Angle θ_U
30 dB JNR Jammers at 15, 50 Degrees

Figure 1.12a: Expanded Antenna Pattern for Weight Vector Calculated for 0.1° (= 5.1° - 5°) Mismatch

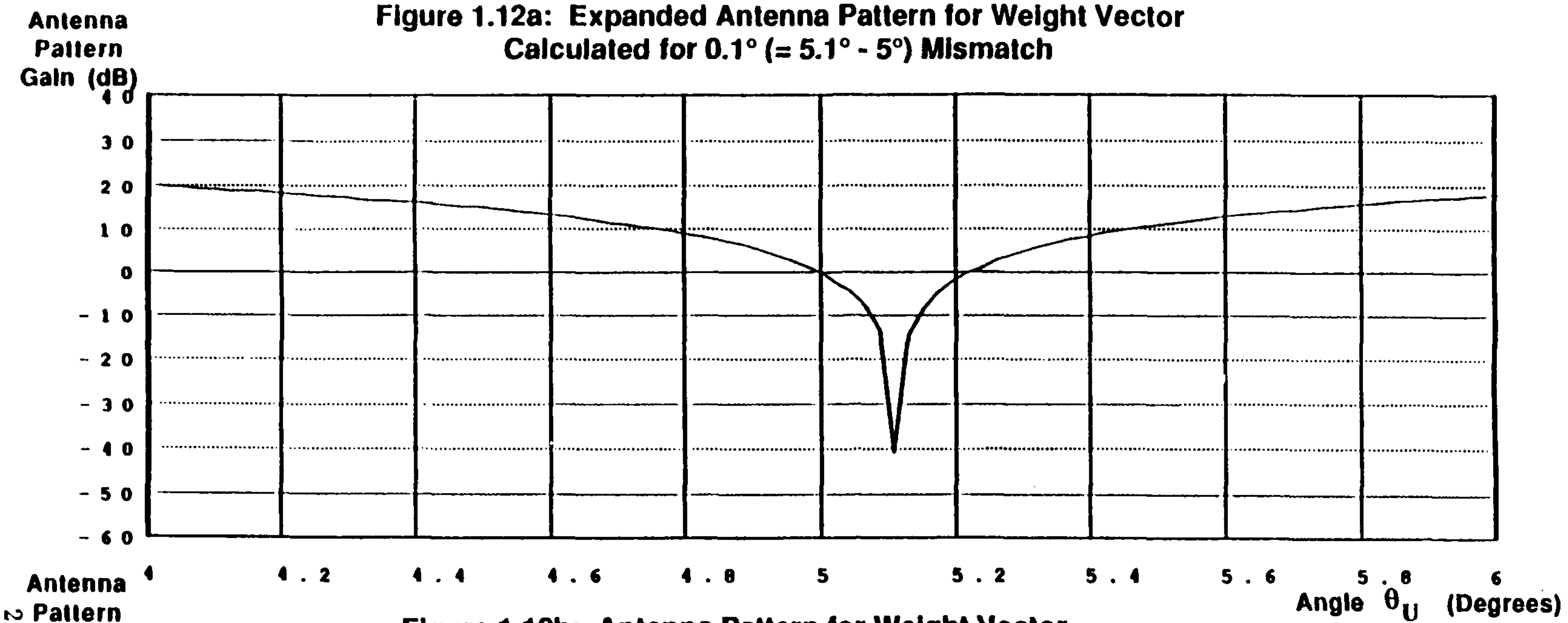


Figure 1.12b: Antenna Pattern for Weight Vector Calculated for 0.1° (= 5.1° - 5°) Mismatch

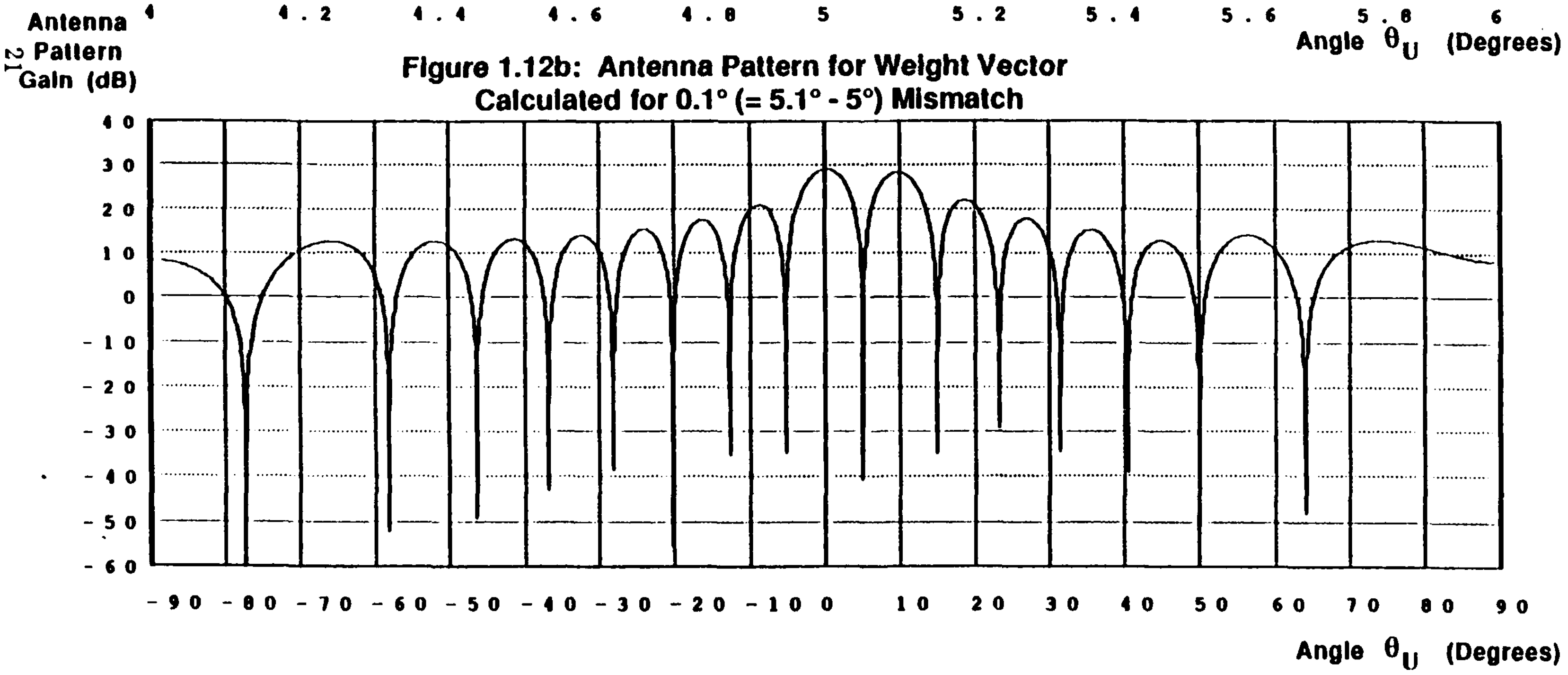
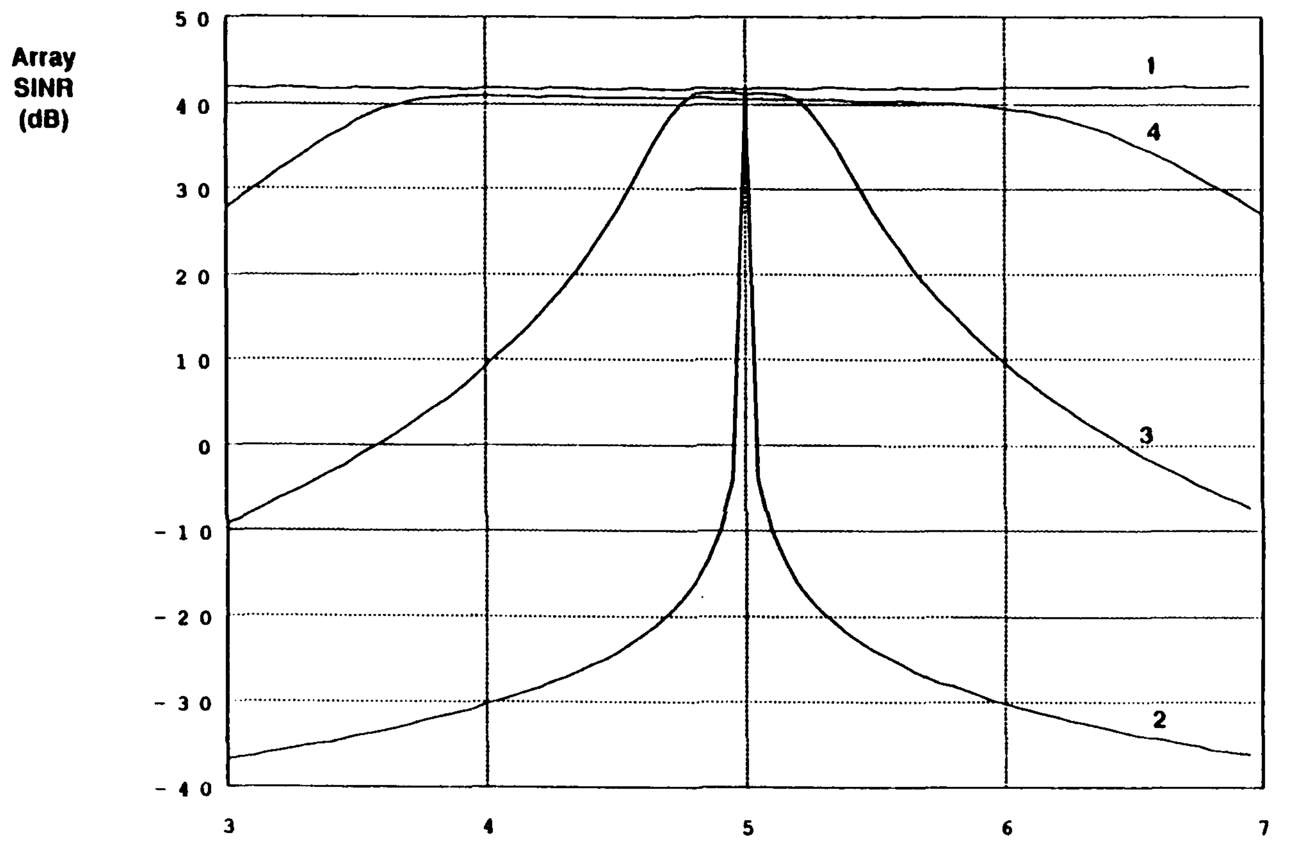


Figure 1.13: Use of New Algorithm to Reduce Impact of Pointing Error



CURVES:
1. Baseline SINR for Steering Vector Matched to User Direction.
2. SINR as a Function of User Position for SMI With Unity Gain Constraint at $\theta = 5°$
3. SINR as a Function of User Position Using New Algorithm with Two-Point Prefilter.
4. SINR as a Function of User Position Using New Algorithm with Three-Point Prefilter.

User Angle θ_U (Degrees)

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

is comparable to that of some conventional algorithms with fixes. When corresponding fixes are added to the new class of algorithms, they typically exhibit improved performance (or reduced sensitivity).

5. The sensitivity of the new class of algorithms to the foregoing types of errors under transient conditions is far superior to that of many conventional nulling algorithms with fixes. For example with regard to the foregoing scenario and a 30 dB SNR user, Figure 1.14 contrasts the user mismatch sensitivity of conventional SMI with no fixes (Curve 2), and with derivative-constrained fixes [5][11] (Curves 3 and 4), to that of steady-state SMI with no user present (Curve 1) after 100 snapshots. The strong adverse impact of the user signal is evident. Figure 1.15 presents analogous results for an example of the new class of algorithms. Clearly user mismatch sensitivity is greatly reduced.

6. Presence of the user signal in the data used for adaptation tends to greatly magnify the effects of uncompensated channel gain and phase errors. As discussed in Chapter 5, the new class of algorithms yields substantial performance benefits in the face of these type of hardware errors.

1.3.4 Example System Architecture

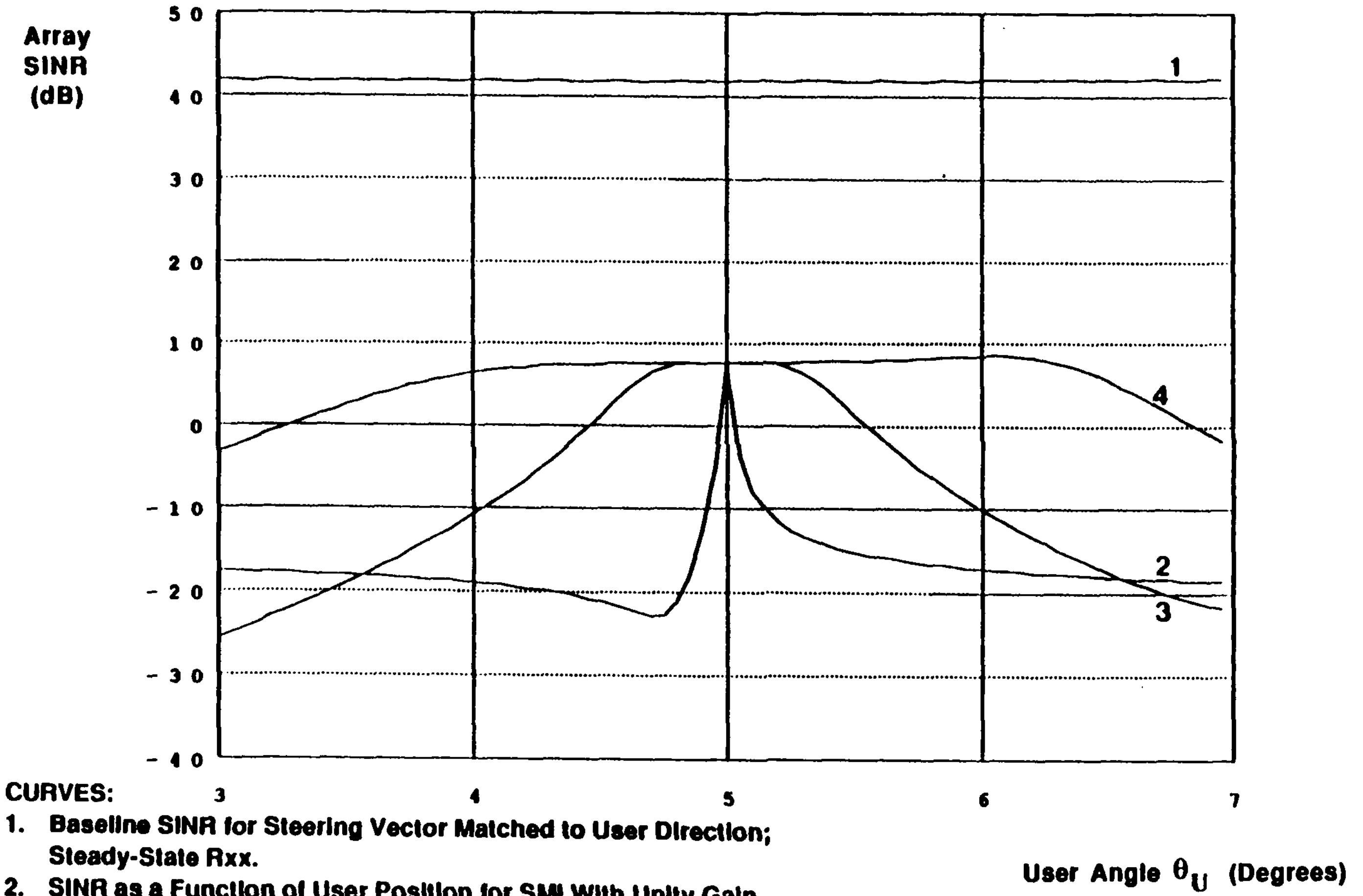
The main results with regard to an example system architecture are:

1. High-level design of a hybrid digital/system architecture which determines the nulling weights in a flexible digital computation module and applies the weights with analog devices.

2. High-level design of subsystems including beamforming network, correlation alternatives, digital processing module, and calibration network.

3. Development of an on-board calibration scheme which compensates for path differences between the weight determination and weight application channels.

Figure 1.14: Illustration of Transient Performance of Derivative-Constrained SMI With User Present and Pointing Error

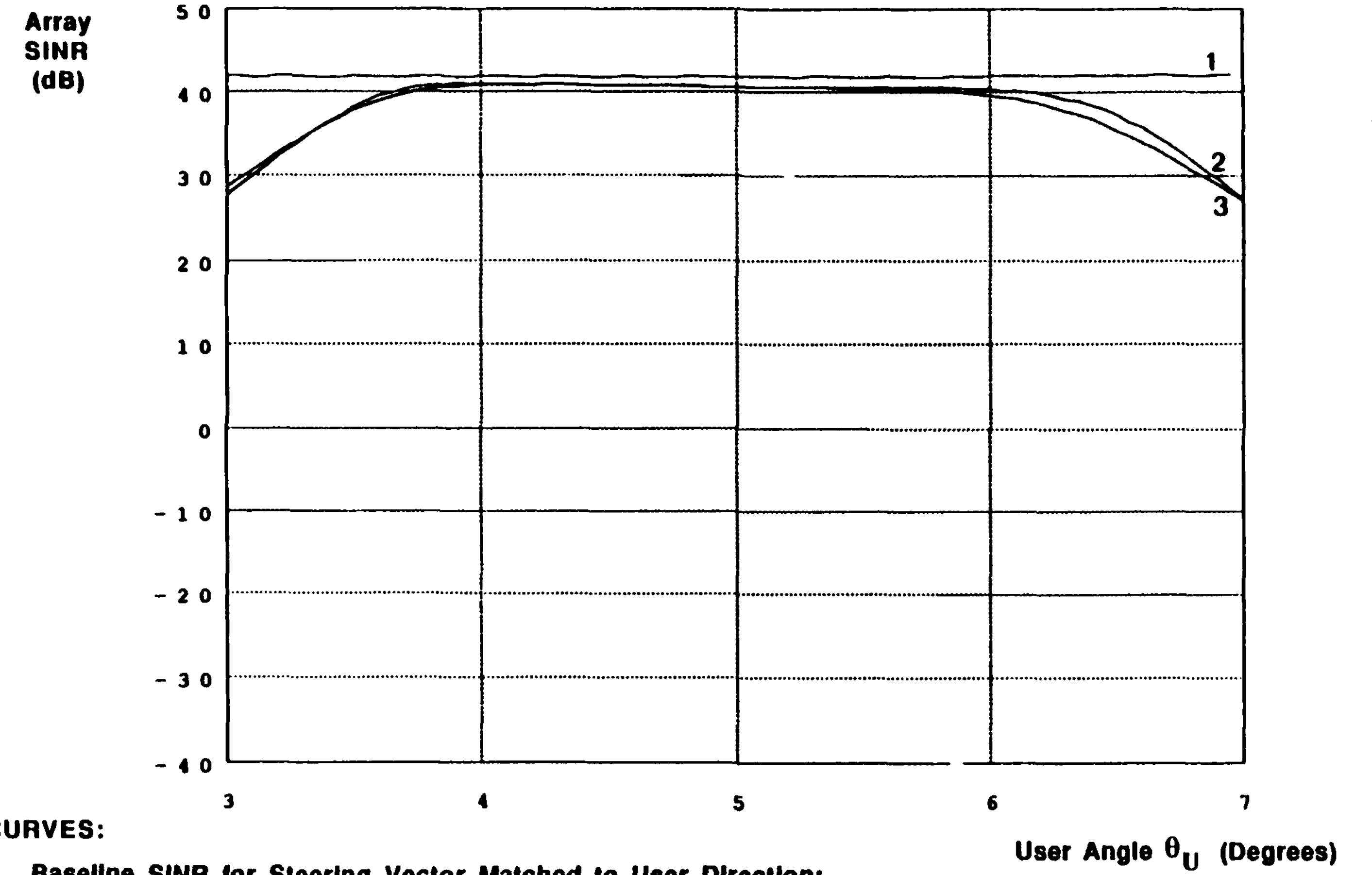


CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction; Steady-State Rxx.
2. SINR as a Function of User Position for SMI With Unity Gain Constraint at $\theta = 5°$; $\hat{R}xx$ Computed Using 100 Snapshots.
3. SINR as a Function of User Position for SMI With Unity Gain and Zero First Derivative Constraint at $\theta = 5°$; $\hat{R}xx$ Computed Using 100 Snapshots.
4. SINR as a Function of User Position for SMI With Unity Gain and Zero First and Second Derivative Constraints at $\theta = 5°$; $\hat{R}xx$ Computed Using 100 Snapshots.

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

Figure 1.15: Illustration of Steady-State and Transient Performance of New Algorithm (Three-Point Prefilter) with User Present and Pointing Error



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction; Steady-State Rxx.
2. SINR as a Function of User Position Using Three-Point Pre-Filter Algorithm; Steady-State Rxx.
3. SINR as a Function of User Position Using Three-Point Pre-Filter Algorithm; 100 Snapshot $\hat{R}xx$.

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

1.4 Organization of the Report

The report is organized into 6 chapters. A large number of examples are included to support major points. A brief description of the contents of each chapter is as follows:

Chapter 2

This chapter sets the stage for the remainder of the report. The chapter, which contains no new results, presents basic assumptions, and reviews relevant results from the literature. Our new results are contained in Chapters 3-6. Section 2.2 presents the assumptions for narrowband arrays. Section 2.3 presents corresponding assumptions for broadband arrays. Section 2.4 reviews conventional power-minimization algorithms for selecting adaptive weights; the section addresses both closed-loop and open-loop algorithms including that due to Applebaum and Howells, LMS and SMI. Finally, Section 2.5 reviews the important topic of beamspace processing, and includes discussion of the GSC configuration and a recent variant due to Gabriel.

Chapter 3

This chapter addresses the issue of designing an open-loop processor that can rapidly produce a set of effective nulling weights in a non-stationary environment with one or more user signals present. Section 3.2 documents the adverse impact of user signal(s) upon the transient response of SMI and related algorithms. Section 3.3 describes the new class of algorithms that accelerate the convergence rate by filtering out the user signal prior to weight calculation. Section 3.4 presents examples illustrating the comparative performance of the algorithms for one-dimensional arrays. Section 3.5 presents examples for two-dimensional arrays.

Chapter 4

This chapter examines alternative nulling techniques applicable to broadband communications problems. It is assumed that the objective is to extract, by spatial filtering, a desired broadband user signal from a broadband jamming environment, and to do so without significantly distorting the user signal. Section 4.2 assesses a variety of techniques for obtaining enhanced broadband performance from narrowband arrays, including related techniques for producing multiple nulls, clustered nulls and elevation nulls for both one and two-dimensional arrays. Section 4.3 addresses techniques applicable to broadband arrays, and demonstrates how multiple nulls and clustered nulls also can be produced for these arrays.

Chapter 5

This chapter assesses the performance of the new class of algorithms of Section 3.3 under non-ideal operating conditions which include: pointing errors and hardware errors. Section 5.2 reviews the pointing error problem for the user-present case, and compares the performance of the new algorithms to a number of conventional approaches. Section 5.3 reviews the channel hardware error problem for the user present case and shows that the effects are similar to those of pointing errors; comparison of the new algorithms with conventional algorithms are presented. The conventional approaches addressed in Sections 5.2 and 5.3 include utilization of linear constraints, norm constraints, and thinned beam methods.

Chapter 6

This chapter describes an example architecture for an on-board nulling system for wideband signals. Section 6.2 describes the high-level architecture design which seeks to minimize hardware size, weight, complexity and power consumption. Section 6.3 discusses the design rationale. Section 6.4 describes the subsystems which comprise the system architecture.

Bibliography

[1] "Advanced Nulling Techniques: Phase I Final Report", Atlantic Aerospace Electronics Corporation (Formerly Pollard Road Inc.), AF Contract Number F04701-85-C-0078, Mar. 1986.

[2] L.J. Griffiths and C.W.Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming", *IEEE Trans. Antennas and Propagat.*, Vol. AP-30, No. 1, pp. 27-34, Jan. 1982.

[3] "Spatial/Spectral Filtering with Linearly Constrained Minimum Variance Beamformers", *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-35, No. 3, pp. 249-266, Mar. 1987.

[4] I.S. Reed, J.D. Mallet and L.E. Brennan, "Rapid Convergence Rate in Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-10, No.6, pp. 853-863, Nov. 1974.

[5] J.E. Hudson, *Introduction to Adaptive Arrays,* Chap. 5, Peter Peregrinus Ltd, New York and London, 1981.

[6] S. Applebaum and D. Chapman, "Adaptive Arrays with Mainbeam Constraints", *IEEE Trans. Antennas and Propagat.*, Vol. AP-24, No. 5, pp. 650-661, Sept 1976.

[7] J. Maksym, "A robust formulation of an optimum cross-spectral beamformer for line arrays", *J. Acoust. Soc. Am.*, Vol. 65, pp.971-975, 1979.

[8] J. Griffiths and J. Hudson, "An introduction to adaptive processing in a passive sonar system", in *Aspects of Signal Processing,* Proceedings of the NATO Advanced Study on signal processing and underwater acoustics, Porto Venere, Italy, 1976, (Reidel, 1977).

[9] W. White, "Artificial Noise in Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-14, No. 2, pp. 380-387, Mar. 1978.

[10] W.F. Gabriel, "Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems", *IEEE Trans. Antennas and Propagat.*, Vol. AP-34, No. 3, pp. 291-300, Mar. 1986.

[11] M. Er and A. Cantoni, "Derivative Constraints for Broadband Element Space Antenna Array Processors", *IEEE Trans. Acoust., Speech, Sig Proc.*, Vol. ASSP-31, pp. 1378-1393, Dec. 1983.

Chapter 2

Review of Prior Work

Contents

List of Figures

2.1 Introduction

The purpose of this chapter is to set the stage for the remainder of the report. This chapter, which contains no new results, presents basic assumptions, and reviews relevant results from the literature. Our new results are contained in Chapters 3-7.

Section 2.2 presents the assumptions for narrowband arrays. Section 2.3 presents corresponding assumptions for broadband arrays. Section 2.4 reviews conventional power-minimization algorithms for selecting adaptive weights; the section addresses both closed-loop and open-loop algorithms including that due to Applebaum and Howells, LMS and SMI. Finally, Section 2.5 reviews the important topic of beamspace processing, and includes discussion of the GSC configuration and a recent variant due to Gabriel.

2.1.1 Generic Adaptive Processor

Figure 2.1 illustrates a generic adaptive processor. It is assumed that a set of desired and undesired signals simultaneously, are incident upon an array of sensors. For convenience, we designate the desired signals as *user signals*, and designate the unwanted signals as *interference* or *jamming signals*. All signals are assumed to emanate from spatially discrete, distant emitters. The objective of the adaptive processor is to linearly combine the sensor outputs x_1, x_2 etc. into an *array output* y that contains strong replicas of the user signals, but contains minimal interference. Thus, the function of the processor is to perform *spatial filtering*, passing the user signals while rejecting the interference (or jammer) signals.

2.1.2 Convolutional Arrays

For purposes of this report, we assume that the sensors are placed within *convolutional arrays*. A convolutional array is defined as a composite array which can be constructed from two component arrays designated as the *sub-array* and the *reference array*. The convolutional array is constructed by translating replicas of the sub-array to positions specified by the reference array. For example, the sensor array of Figure 2.2c is a two-dimensional convolutional array since it can be constructed by translating replicas of the sub-array shown in Figure 2.2a so that the reference points on the sub-arrays (marked by X's) coincide with the reference points depicted in Figure 2.2b. Figures 2.3a and 2.3b depict further examples of convolutional arrays.

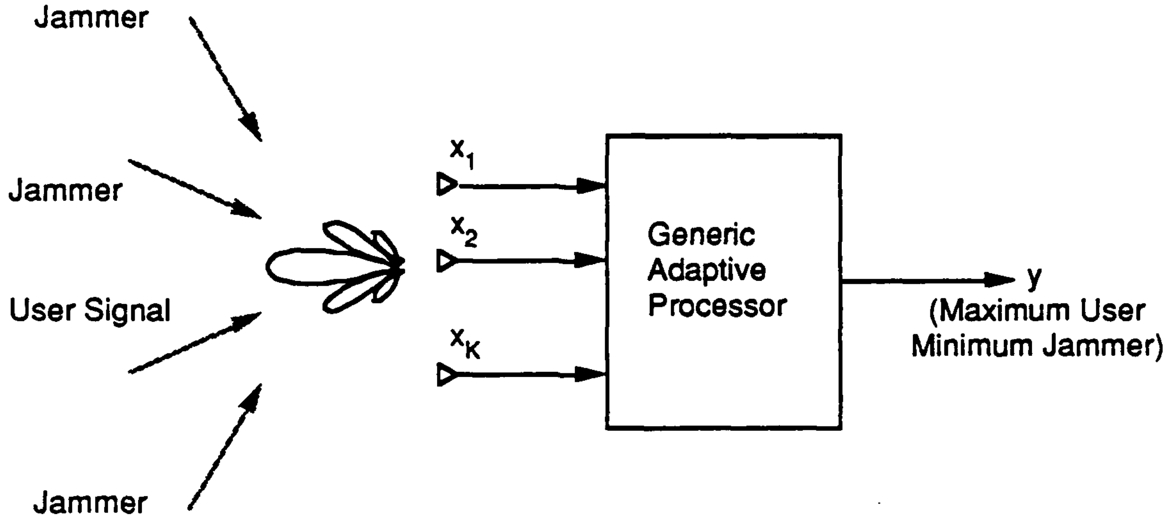


Figure 2.1: Generic Adaptive Processor

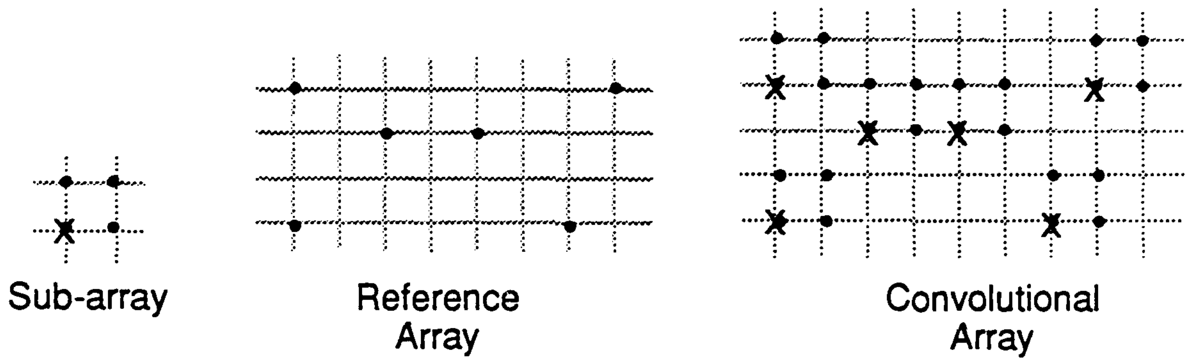


Figure 2.2: Structure of a Convolutional Array

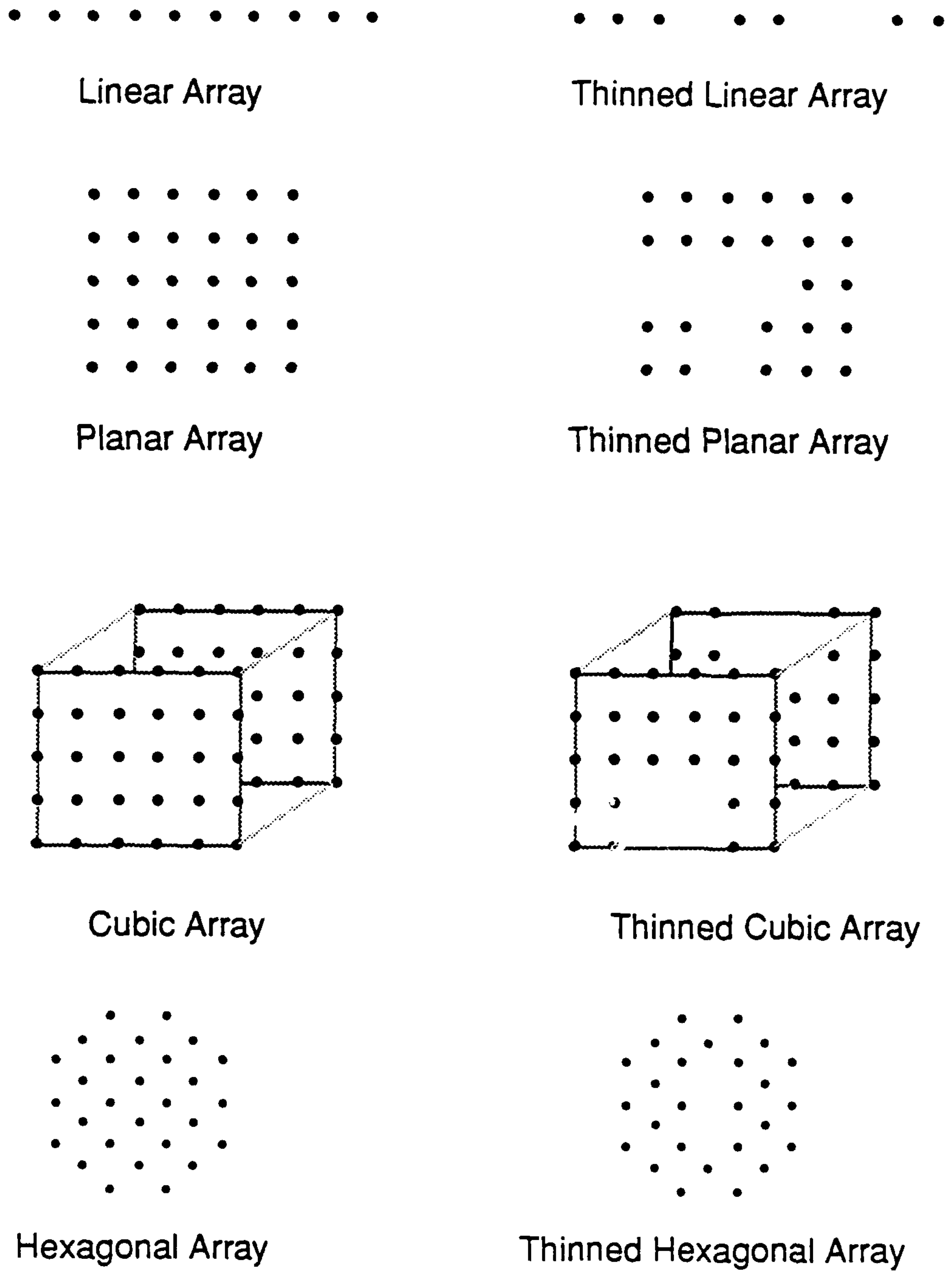


Figure 2.3: Examples of Convolutional Arrays

Clearly, all one, two, and three-dimensional arrays with uniformly spaced elements are convolutional arrays. While many practical arrays can be modelled as convolutional arrays, it should be noted that not all arrays are convolutional.

The antenna patterns of convolutional arrays exhibit the well-known product property of Fourier transform theory [1]. Namely, the Fourier transform of the convolution of two signals is the product of the Fourier transform of each signal. Because of the duality between array weights/signal values, and beampatterns/Fourier transforms, the property is also applicable to convolutional arrays. Specifically, if the sensor signals of the sub-arrays are processed by identical weight-and-sum processors to create sub-array output signals, and these signals are then processed by a follow-on processor to create the output signal for the composite array, then the far-field antenna pattern for the composite array is the product of that of the weighted sub-array and that of the weighted reference array.

2.2 Narrowband Arrays

We will designate a nulling problem as a narrowband problem if the user signals are sufficiently narrowband that they can be treated as tones. More precisely, we will designate a problem as narrowband if

$$(\Delta f)T \ll 1 \tag{2.1}$$

where Δf denotes the bandwidth of the user signals, and T denotes the maximum signal propagation time across the array.

In narrowband problems, the adaptive processor will consist of a *weight-and-sum processor* as illustrated in Figure 2.4. That is, the output of sensors $1 \cdot \cdot K$ are multiplied by complex gains (or weights) $w_1^*, w_2^* \cdot \cdot w_K^*$ and the weighted signals then are summed to form the array output y. The objective of the adaptive algorithm is to select the weights $w_1^*, w_2^* \cdot \cdot w_K^*$ such that the user signals are strongly present in y, and the jammer signals are minimally present. Thus, if the (receiving) antenna pattern is constructed for the array at the frequency of the user signal, then the gain should be large in the directions of the users, and the pattern should exhibit nulls on or near the jammers.

2.2.1 Basic Assumptions and Notation for Narrowband Arrays

For purposes of assessing the performance of narrowband arrays, we make the following assumptions:

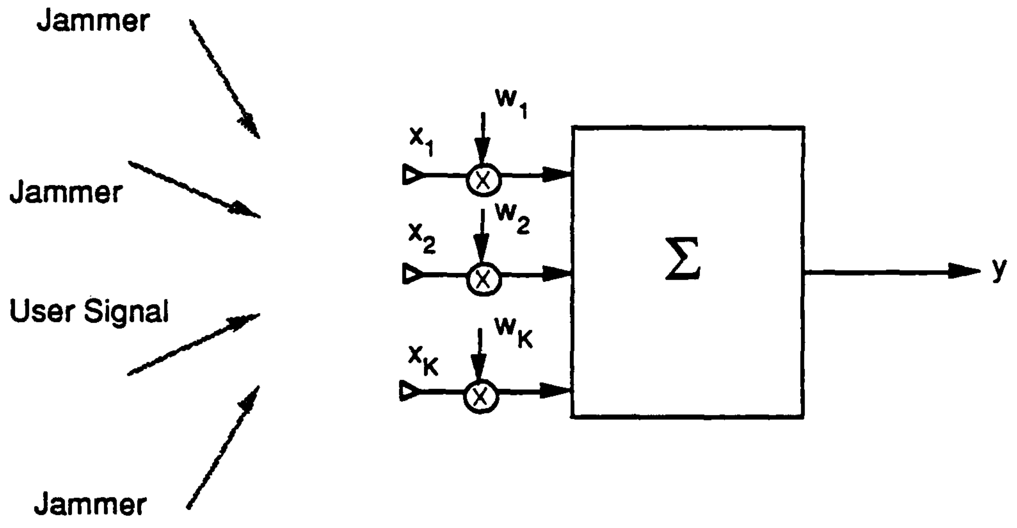


Figure 2.4: Narrowband Weight-and-Sum Processor

N1: The receiving array consists of K distinct sensors having (identical) omni-directional antenna patterns.

N2: The sensors are placed in a convolutional array.

N3: The array typically is illuminated by waves from one or more users, and from a number J of spatially discrete jammers.

N4: The user positions are known; the jammer positions are unknown.

N5: The complex signal x_k output by the k^{th} sensor is the sum of the complex signals incident upon the sensor from all emitters plus additive noise. That is,

$$x_k = \sum_i a_i u_{k,i} + \epsilon_k \tag{2.2}$$

where $u_{k,i}$ denotes the complex signal induced in the k^{th} sensor by an incident wave of unit amplitude from the i^{th} emitter, a_i denotes the complex amplitude of the wave from the i^{th} emitter, ϵ_k denotes the complex noise signal added by the sensor, and

$$E\{\epsilon_j^* \epsilon_k\} = \sigma^2 \delta_{jk} \tag{2.3}$$

where $E\{\}$ denotes expectation, and δ is the Kronecker delta function.

N6: The outputs of all sensors are sampled at identical times. The vector

$$\vec{x}(n) = [x_1(n), x_2(n), \cdots x_k(n)]^T \tag{2.4}$$

of samples for the n^{th} sampling instant is called an *array snapshot.* Clearly,

$$\vec{x}(n) = \sum_i a_i(n) \vec{u}_i + \vec{\epsilon}(n) \tag{2.5}$$

where $\vec{u}_i$ denotes the vector of signals induced by a unit wave from the i^{th} emitter:

$$\vec{u}_i = [u_{1,i}, u_{2,i}, \cdots u_{K,i}]^T \tag{2.6}$$

$a_i(n)$ denotes the complex amplitude of the wave from the i^{th} emitter at the n^{th} sampling instant, and $\vec{\epsilon}(n)$ denotes the vector of noise samples:

$$\epsilon(n) = [\epsilon_1(n), \epsilon_2(n), \cdots \epsilon_K(n)]^T \tag{2.7}$$

N7: The sensor outputs $x_1(n) \cdot\cdot x_K(n)$ are multiplied by complex weights $w_1^* \cdot\cdot w_K^*$, and the weighted signals are summed to form the array output $y(n)$ for the n^{th} sampling time. (See Figure 2.4). That is,

$$y(n) = \sum_{k=1}^{K} w_k^* x_k(n) \tag{2.8}$$

N8: The vector

$$\vec{w} = [w_1, w_2, \cdot\cdot w_K]^T \tag{2.9}$$

is called the weight vector. From (2.5), (2.8) and (2.9), it is clear that the array output $y(n)$ can be written as the inner product

$$y(n) = \vec{w}^H \vec{x}(n) \tag{2.10}$$

the superscript H denotes conjugate transpose.

N9: Provided the amplitudes $a_i(n)$ and $\epsilon_k(n)$ are samples of stationary processes, the ensemble average power p out of the array is

$$p \equiv E\{|y(n)|^2\} = \vec{w}^H R_{xx} \vec{w} \tag{2.11}$$

where $E\{\}$ denotes expectation, R_{xx} denotes the covariance matrix

$$R_{xx} \equiv E\{\vec{x}(n)\vec{x}^H(n)\} \tag{2.12}$$

N10: The time average power $\hat{p}$ out of the array is

$$\hat{p} \equiv \frac{1}{N} \sum_{n=1}^{N} |y(n)|^2 = \vec{w}^H \hat{R}_{xx} \vec{w} \tag{2.13}$$

where $\hat{R}_{xx}$ denotes the sample covariance matrix

$$\hat{R}_{xx} = \frac{1}{N} \sum_{n=1}^{N} \vec{x}(n)\vec{x}^H(n) \tag{2.14}$$

2.3 Broadband Arrays

We will designate a nulling problem as broadband if the bandwidth Δf of the user signals is sufficiently large that condition (2.1) is not satisfied.

In broadband problems, we will assume that the adaptive processor of Figure 2.1 is a filter-and-sum processor as illustrated in Figure 2.5. Specifically, the output of typical k^{th}

sensor is processed by a linear time-invariant filter having the transfer function $H_k(\omega)$ (as opposed to a constant gain factor), and the filtered sensor outputs are summed to produce the array output y. The objective of the adaptive algorithm is to design the K filters such that the user signals are strongly present without distortion in y, and that the jamming signals are minimally present.

One method for performing broadband nulling is to decompose the broadband problem into a number of narrowband problems. This can be accomplished by realizing each broadband filter in Figure 2.5 as a bank of L narrowband filters with independent (complex) gain controls, as shown in Figure 2.6. The gains across the array within each sub-band can be selected via a narrowband algorithm so as to pass the user signals, and to reject the jammer signals. The broadband user signals then will be strongly present in the array output y, and the jammer signals will be minimally present.

An alternate realization for the broadband filters consists of a transversal filter as illustrated in Figure 2.7. The essential properties of this filter are suggested by the Discrete Fourier Transform Theorem. Specifically, by suitably selecting the L complex weights $w^*_{1,k}, w^*_{2,k}, \cdots w^*_{L,k}$, this filter can be made to realize prescribed complex gains at L equally spaced frequencies within the (one-sided) user bandwidth Δf provided the delay τ between successive taps satisfies

$$(\Delta f)\tau = 1 \tag{2.15}$$

Thus, in principle, a transversal filter with L taps can provide filtering performance similar to that of a filter bank having L sub-bands.

For purposes of this report, we assume that the broadband filters in Figure 2.5 are realized by transversal filters as illustrated in Figure 2.7. The broadband adaptive processor then takes the form shown in Figure 2.8.

2.3.1 Basic Assumptions and Notation for Broadband Arrays

For purposes of assessing the performance of broadband arrays, we make the following assumptions. The notation introduced in the assumptions is such that the same expressions for array output signal, power, constraints, etc., apply to both broadband and narrowband problems.

B1-B4: Identical to N1-N4.

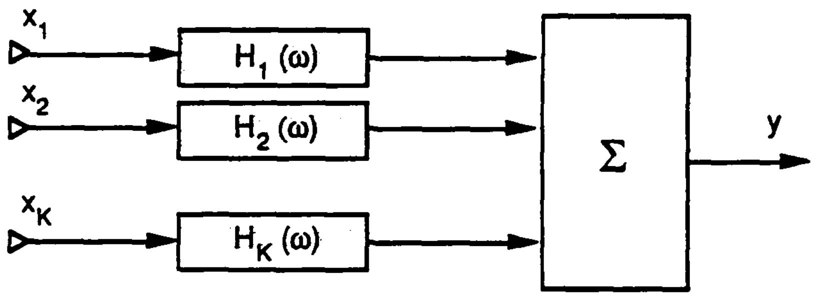


Figure 2.5: Broadband Filter-and-Sum Processor

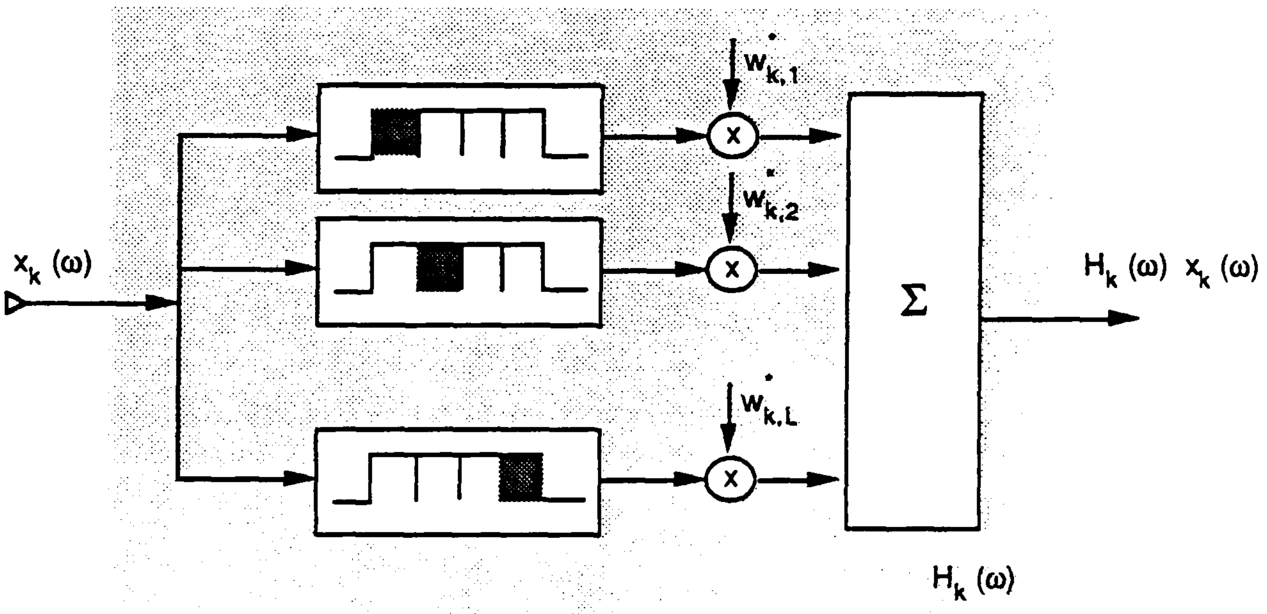


Figure 2.6: Realization of $H_k(\omega)$ as a Bank of Narrowband Filters

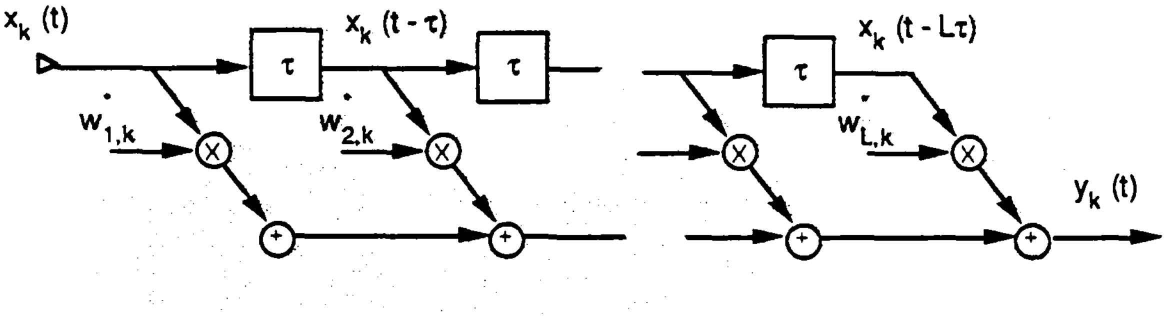


Figure 2.7: Transversal Filter

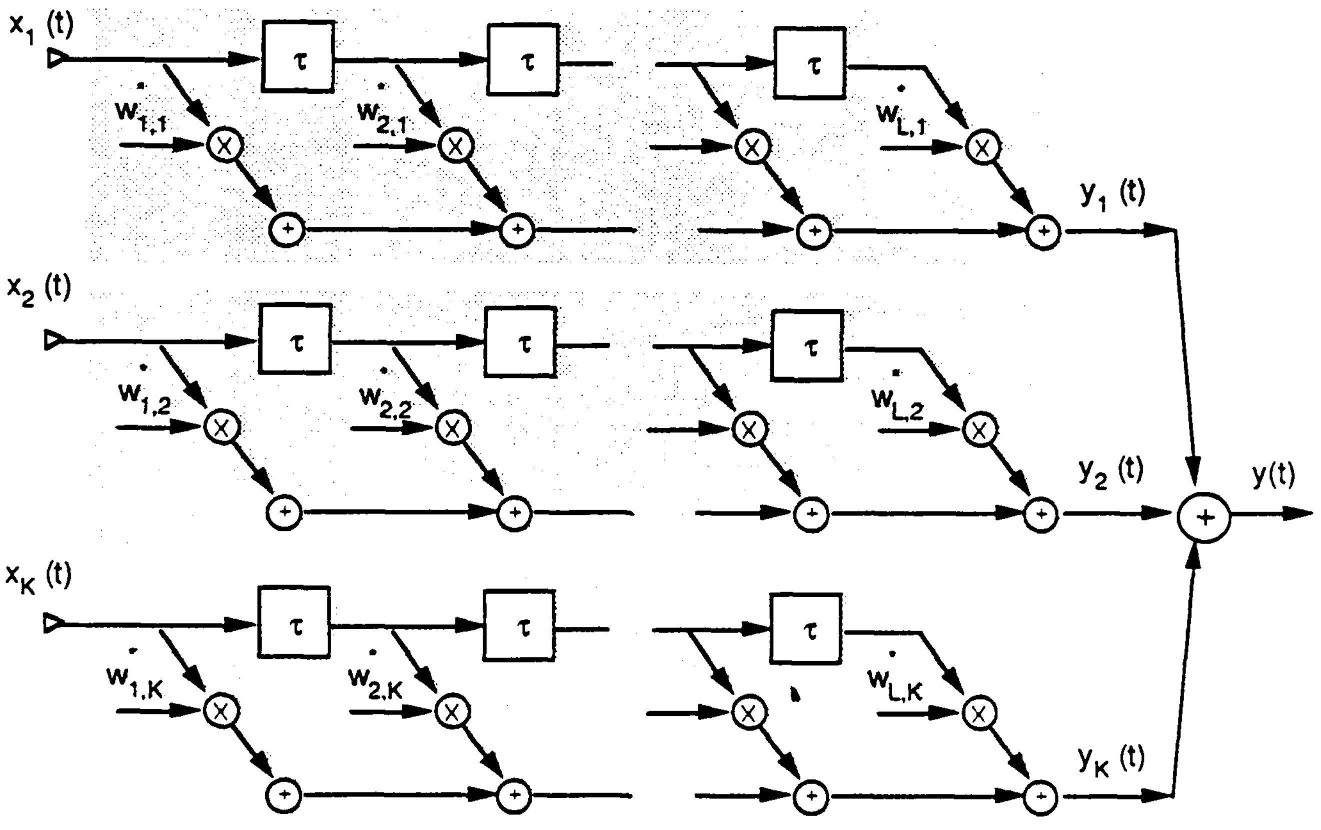


Figure 2.8: Broadband Adaptive Processor

B5: The signal output by the k^{th} sensor is of the form

$$x_k(t) = \sum_i a_i u_i(t - \vec{r}_k \cdot \vec{v}_i/c) + n_k(t) \tag{2.16}$$

where t denotes time, a_i and $u_i()$ respectively denote the amplitude and waveform of the signal from the i^{th} emitter, $\vec{r}_k$ is the radius vector from a reference point O to the k^{th} sensor, $\vec{v}_i$ is a unit vector directed from point O to the i^{th} source, c is the speed of light, and $n_k(t)$ is an additive noise waveform.

The waveform $u_i()$ is a sample function for a white (one-sided) band-limited random process having the auto-correlation function

$$r_s(\varsigma) = E\{u_i(t)u_i^*(t-\varsigma)\} = \frac{1}{2\pi}\int_{\omega_l}^{\omega_h} S(\omega)e^{j\omega\varsigma}d\omega \tag{2.17}$$

where

$$S(\omega) = \sigma_s^2/(\omega_h - \omega_l) \tag{2.18}$$

The waveforms are uncorrelated from emitter to emitter so that

$$E\{u_i(t)u_j^*(t-\varsigma)\} = 0 \quad i \neq j \tag{2.19}$$

The noise waveforms $n_k(t)$ are sample functions for a white band-limited random process having the auto-correlation function

$$r_n(\varsigma) = E\{n_k(t)n_k^*(t-\varsigma)\} = \frac{1}{2\pi}\int_{\omega_h}^{\omega_h} N(\omega)e^{j\omega\varsigma}d\omega \tag{2.20}$$

where

$$N(\omega) = \sigma_n^2/(\omega_h - \omega_l) \tag{2.21}$$

B6: The noise waveforms are uncorrelated from sensor to sensor, and also are uncorrelated with the emitter waveforms so that

$$E\{n_j(t)n_k^*(t-\varsigma)\} \equiv 0 \quad j \neq k \tag{2.22}$$

and

$$E\{n_j(t)u_i^*(t-\varsigma)\} \equiv 0 \tag{2.23}$$

B7: The outputs of all sensors are sampled at the (one-sided) Nyquist rate, or with the intersample spacing of

$$\tau = 2\pi/(\omega_h - \omega_l) \tag{2.24}$$

The vector

$$\vec{x}(n\tau) = [x_1(n\tau), x_2(n\tau), \cdots x_K(n\tau)]^T \tag{2.25}$$

is called the instantaneous snapshot vector. The vector

$$\vec{x}(n) = [\vec{x}^T(n\tau), \vec{x}^T((n-1)\tau), \cdots \vec{x}^T((n-L+1)\tau)]^T \tag{2.26}$$

consisting of L consecutive instantaneous snapshots is called the *stacked array snapshot.* The K sensor signals are input into a bank of K tapped delay lines with inter-tap spacing given by (2.24) as illustrated in Figure 2.8. Note that if the elements of the instantaneous snapshot vector $\vec{x}(n\tau)$ appear at the first taps of the delay lines, then the elements of the instantaneous snapshot vector $\vec{x}((n-l+1)\tau)$ appear at the l^{th} tap.

B8: The output of the delay line taps are processed by a weight-and-sum processor to produce the array output $y(n)$ as illustrated in Figure 2.8. The weight applied to the signal at the l^{th} tap of the k^{th} delay line is w_{lk}^*. The vector

$$\vec{v}_l = [w_{l1}, w_{l2}, \cdots w_{lK}]^T \tag{2.27}$$

is called the l^{th} *snapshot weight vector.* The vector

$$\vec{w} = [\vec{v}_1^T, \vec{v}_2^T, \cdots \vec{v}_L^T]^T \tag{2.28}$$

consisting of the L snapshot weight vectors is called the *stacked weight vector.* The array output is

$$y(n) = \vec{w}^H \vec{x}(n) \tag{2.29}$$

B9-B10: Identical to N9 and N10. The matrix R_{xx} for a broadband array is referred to as the *stacked covariance matrix.*

2.4 Weight Selection Via Power Minimization

The classical approach to selecting the weights of the adaptive processors illustrated in Figures 2.4 and 2.8 is to choose the weight vector $\vec{w}$ so as to minimize the expected array output power

$$p = \vec{w}^H R_{xx} \vec{w} \tag{2.30}$$

subject to one or more constraints on the weight vector. In the simplest case, the constraint is that the array gain to a single user be a prescribed value; for example

$$1 = \vec{w}^H \vec{u}_0 = \vec{u}_0^H \vec{w} \tag{2.31}$$

where $\vec{u}_0$ denotes the vector

$$\vec{u}_0 = [u_{1,0}, u_{2,0}, \cdots u_{K,0}]^T \tag{2.32}$$

of signals induced by a unit waveform the user. Minimization of (2.30) subject to (2.31) shows that the optimum weight vector satisfies the Wiener-Hopf equation

$$R_{xx}\vec{w} = \mu \vec{u}_0 \tag{2.33}$$

so that

$$\vec{w} = \mu R_{xx}^{-1} \vec{u}_0 \tag{2.34}$$

where μ denotes the constant

$$\mu = 1/(\vec{u}_0^H R_{xx}^{-1} \vec{u}_0) \tag{2.35}$$

The corresponding minimum power is

$$p = 1/(\vec{u}_0^H R_{xx}^{-1} \vec{u}_0) \tag{2.36}$$

(See Reference [3]).

In the case of Q linear constraints on $\vec{w}$, the single equation (2.31) is replaced by a matrix equation of the form

$$C^H \vec{w} = \vec{f} \tag{2.37}$$

where C^H is a $Q \times L$ matrix of constants, and $\vec{f}$ is a $Q \times 1$ vector of constants. Equation (2.37) can incorporate a variety of useful constraints including gain requirements for multiple users at a single frequency, gain requirements to users at multiple frequencies, antenna-pattern derivative control, etc. Minimization of (2.30) subject to the more general constraint condition (2.37) yields for the optimum weight vector

$$\vec{w} = R_{xx}^{-1} C (C^H R_{xx}^{-1} C)^{-1} \vec{f} \tag{2.38}$$

The corresponding minimum power is

$$p = \vec{f}^H (C^H R_{xx}^{-1} C)^{-1} \vec{f} \tag{2.39}$$

(See Reference [6]). Clearly (2.38) and (2.39) reduce to (2.34), (2.35) and (2.36) for the single constraint case (2.37) in which $C = \vec{u}_0$. Comparison of (2.38) with (2.34) shows that the effect of $Q > 1$ constraints is to replace the forcing vector $\vec{u}_0$ in (2.34) by the constraint-space vector $C(C^H R_{xx}^{-1} C)^{-1} \vec{f}$.

2.4.1 Classical Nulling Algorithms

While Equations (2.38) and (2.39) specify optimum weight vectors for the adaptive processor given a stationary environment, the covariance matrix R_{xx} typically is not available. Thus, the objective of classical nulling algorithms is to approximate the optimum solutions (2.38) and (2.39) based upon snapshots $\vec{x}(n)$ collected over a finite time interval. The techniques differ according to the manner in which this approximation is performed.

2.4.2 Closed-Loop Algorithms

The classical Applebaum-Howells and Least-Mean-Square (LMS) algorithms [2][3][4] are examples of ***closed-loop techniques*** for selecting the weights of the adaptive processor so as to minimize power. At the outset, those algorithms use a nominal set of weights and then refine the weights over time based upon cross-correlation measurements between the array output y and the (input) array snapshot $\vec{x}(n)$. Thus, the array output y is "fed back" into the algorithm that adjusts the weights. Both algorithms can be viewed as utilizing steepest descent methods to reduce the time-average output power as more data is collected. Given a stationary environment, an approximation of the optimum (power-minimizing) weight set eventually is obtained.

Closed-loop algorithms have the desirable property of minimizing power non-withstanding modest non-ideal effects such as receiver channel misalignment and non-linearity. The processors have the undesirable property of slow convergence in certain scenarios, which can seriously compromise performance in dynamic (non-stationary) environments[9][10].

Applebaum-Howells Processor [2]-[4]

Figure 2.9 depicts the (narrowband) Applebaum-Howells adaptive processor for the case of a single constraint on the weight vector $\vec{w}$. The processor consists of K integrating correlators and a summation operator as illustrated. The outputs of sensors $1 \cdot \cdot K$ are multiplied by complex weights $w_1^*, w_2^* \cdot \cdot w_K^*$, and the results are summed to produced the output y given by (2.8) or (2.10). The integrating correlators determine the weights as illustrated in Figure 2.10. Signals equal to successive elements of the unit arrival vector $\vec{u}_0$ given by (2.6) are applied as reference signals to the differential high-gain amplifiers. The array output y is fed back and correlated with the array snapshot $\vec{x}(n)$ to produce the vector

$$\vec{z} = \overline{\{\vec{x}y^*\}} = \overline{\{\vec{x}\vec{x}^H\}}\vec{w} = (1/\mu)\hat{R}_{xx}\vec{w} \qquad (2.40)$$

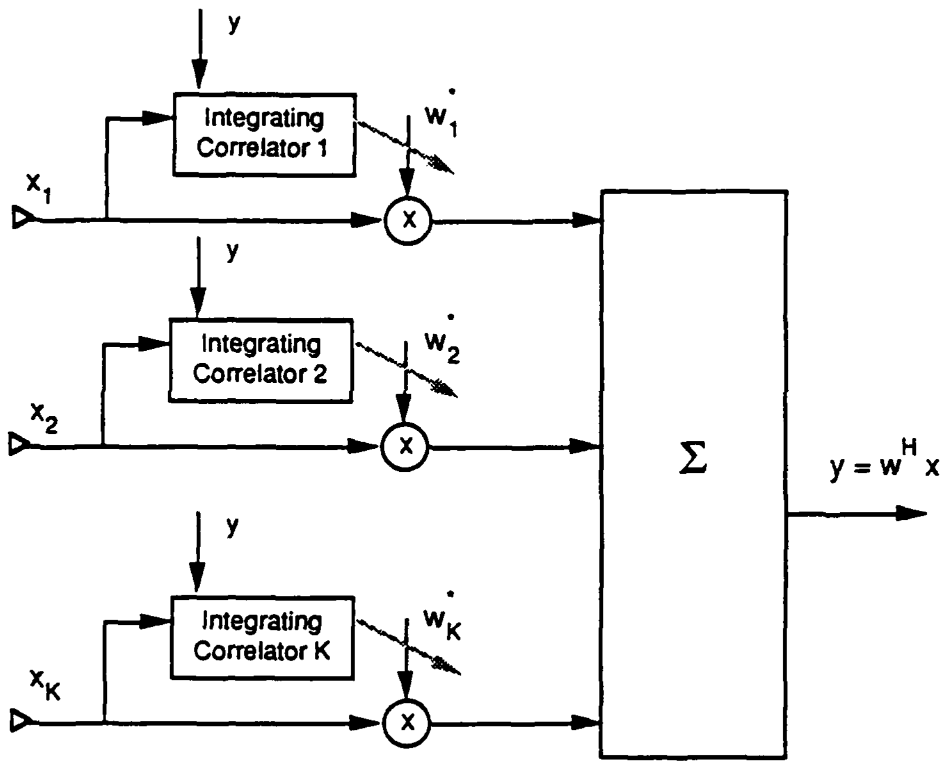


Figure 2.9: Applebaum-Howells Processor

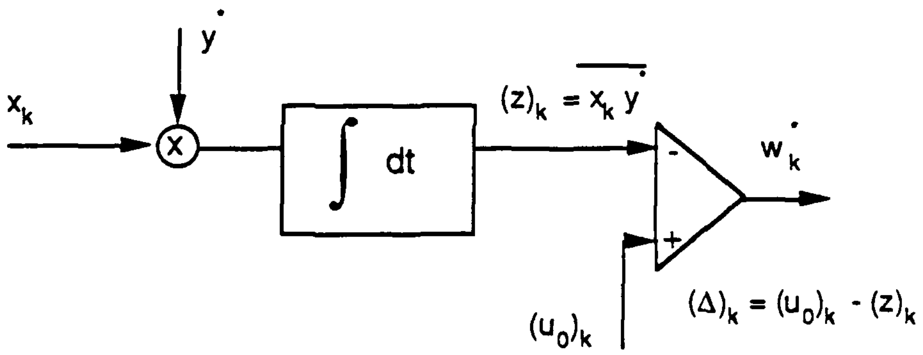


Figure 2.10: The kth Integrating Correlator

where $\hat{R}_{xx}$ denotes the time-average correlation matrix

$$\hat{R}_{xx} = \mu \int \vec{x}\vec{x}^H d\tau \tag{2.41}$$

Thus, the difference (or error) signals applied to the successive high-gain amplifiers are proportional to successive elements of the vector

$$\vec{\Delta} = \vec{u}_0 - \vec{z} = \vec{u}_0 - (1/\mu)\hat{R}_{xx}\vec{w} \tag{2.42}$$

Given a stationary environment, the multi-loop structure relaxes to a nominal behavior in which the error signals applied to the high-gain amplifiers approximately equal zero; that is,

$$\vec{\Delta} \approx 0 \tag{2.43}$$

so that

$$\hat{R}_{xx}\vec{w} \approx \mu\vec{u}_0 \tag{2.44}$$

As a result, the nominal weight vector $\vec{w}$ approximately solves the Wiener-Hopf equation (2.33).

The Applebaum-Howells processor can be implemented either in analog or digital hardware. The processor also can be generalized straightforwardly to the case of $Q > 1$ constraints by means of a pre-processing beamformer. (See Section 2.5.3.) Note that in the multiple constraint case, one cannot simply replace the reference vector $\vec{u}_0$ for the processor by the vector $C(C^H R_{xx}^{-1} C)^{-1}\vec{f}$ of Equation (2.38) since the matrix R_{xx} is unknown.

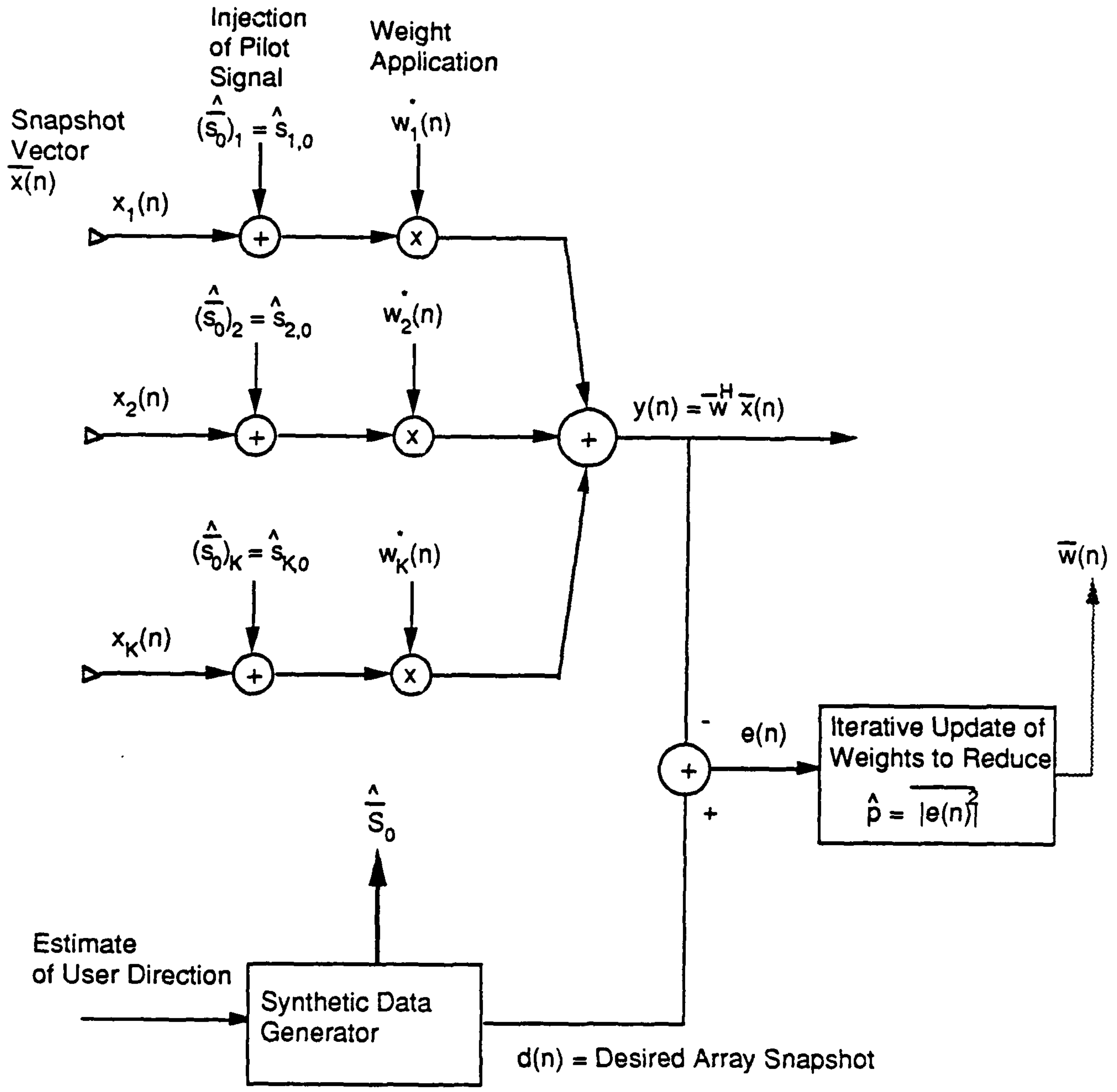
LMS Processor[3]

Figure 2.11 depicts the (narrowband) LMS processor for the case of a single (user-direction) constraint. The essence of the processor algorithm is as follows:

1. A pilot signal $\hat{u}_{k,0}(n)$ is added to each of the sensor outputs. The pilot signals approximate the desired user signals $u_{k,0}(n)$ in relative phase and relative amplitude.

2. The processor generates a signal $d(n)$ that represents the desired output of the array to the injected pilot signals. The signal $d(n)$ is coherent with the $\hat{u}_{k,0}(n)$.

3. By comparing the actual array output $\tilde{y}(n)$ (which includes interference) with $d(n)$, the processor iteratively adjusts the weights $w_1^*, w_2^* \cdot \cdot w_K^*$ with the objective of making



Injection of Pilot Signal
Weight Application
Snapshot Vector $\overline{x(n)}$
$(\hat{\bar{s}}_0)_1 = \hat{s}_{1,0}$
$w_1^*(n)$
$x_1(n)$
$(\hat{\bar{s}}_0)_2 = \hat{s}_{2,0}$
$w_2^*(n)$
$x_2(n)$
$y(n) = \bar{w}^H \bar{x}(n)$
$(\hat{\bar{s}}_0)_K = \hat{s}_{K0}$
$w_K^*(n)$
$x_K(n)$
$\bar{w}(n)$
$e(n)$
Iterative Update of Weights to Reduce $\hat{p} = \overline{|e(n)|^2}$
$\hat{\bar{S}}_0$
Estimate of User Direction
Synthetic Data Generator
d(n) = Desired Array Snapshot

Figure 2.11: LMS Processor

$\tilde{y}(n)$ approximate $d(n)$ insofar as possible in the mean-square sense. That is the processor adjusts the weights with the objective of minimizing the mean square of $d(n) - \tilde{y}(n)$.

A number of devices are available for preventing the injected pilot signals from interfering with the actual user signal[4][5].

Addition of the pilot signals $\hat{u}_{k,0}(n)$ to the sensor outputs results in the vector

$$\hat{\vec{u}}_0 = [u_{\hat{1},0}, u_{\hat{2},0}, \cdots u_{\hat{K},0}]^T \tag{2.45}$$

being added to the actual snapshot $\vec{x}(n)$ to provide a modified snapshot

$$\tilde{\vec{x}}(n) = \vec{x}(n) + \hat{\vec{u}}_0(n) \tag{2.46}$$

The array output becomes

$$\tilde{y}(n) = \vec{w}^H \tilde{\vec{x}}(n) \tag{2.47}$$

The error $e(n)$ between $d(n)$ and $\tilde{y}(n)$ is

$$e(n) = d(n) - \tilde{y}(n) \tag{2.48}$$

and the mean-square error $\varepsilon(n)$ is

$$\varepsilon(n) = E\{|e(n)|^2\} \tag{2.49}$$

The algorithm iteratively adjusts the weight vector $\vec{w}$ with the objective of reducing (2.49). The steepest descent rule for updating $\vec{w}$ so as to reduce (2.49) is

$$\vec{w}(n+1) = \vec{w}(n) - \alpha\vec{\nabla}[\varepsilon(n)] \tag{2.50}$$

where $\vec{\nabla}$ denotes the gradient of (2.49) with respect to differentials in $\vec{w}$, and α is a (small) positive step size. The gradient vector is approximated as follows

$$\vec{\Delta}[\varepsilon(n)] \approx -2\tilde{\vec{x}}(n)e^*(n) \tag{2.51}$$

and the update rule is taken to be

$$\vec{w}(n+1) = \vec{w}(n) + 2\alpha\tilde{\vec{x}}(n)e^*(n) \tag{2.52}$$

Provided the snapshots $\vec{x}(n)$ are samples of a stationary process, the expected value of the solution $\vec{w}(n)$ of (2.52) approaches the solution $\vec{w}$ of the equation

$$R_{\tilde{x}\tilde{x}}\vec{w} = \mu\hat{\vec{u}}_0 \tag{2.53}$$

as $n \to \infty$, where $\hat{\vec{u}}_0$ denotes the processor's estimate of the vector of unit arrivals from the user. The convergence time is proportional to λ_1/λ_K where λ_1 and λ_K denote respectively the largest and smallest eigenvalues of $R_{\tilde{x}\tilde{x}}$.

Furthermore, if $\hat{\vec{u}}_0 = \vec{u}_0$ then the solution (2.53) also solves the Wiener-Hopf equation (2.33) for the case of no pilot signal.

Griffiths[5] has shown that LMS can be modified both for narrowband and broadband problems to remove the requirement for a pilot signal. Frost[6] has extended LMS to incorporate constraints to the form (2.37). Griffiths et.al [7][8] have shown that if appropriate use is made of a beamforming pre-processor, then constraints of the form (2.37) can be incorporated in a particularly simple manner. (See Section 2.5.3.)

2.4.3 Open-Loop Algorithms

Closed-loop algorithms for selecting the weights of the adaptive processor can provide robust interference suppression given a stationary (or slowly changing) environment and long integration times. However, performance may be unsatisfactory in problems where effective interference suppression is required after a short time interval, or in rapidly changing (non-stationary) environments. In such cases, open-loop algorithms can provide viable alternatives.

Open-loop algorithms approximate the solution of the power minimization problem defined by (2.30) and (2.31) by directly solving a system of equations, rather than by constructing a closed-loop processor which eventually migrates to the desired solution. The Sample Matrix Inversion (SMI) introduced by Reed, et. al.[9], is the best known open-loop algorithm. SMI estimates the covariance matrix (2.12) to be the sample covariance matrix (2.14) and then directly solves the *approximate*Wiener-Hopf equation

$$\hat{R}_{xx}\vec{w}(N) = \mu\vec{u}_0 \tag{2.54}$$

to obtain the weight vector

$$\vec{w}(N) = \mu\hat{R}_{xx}^{-1}\vec{u}_0 \tag{2.55}$$

where

$$\mu = 1/(\vec{u}_0^H\hat{R}_{xx}^{-1}\vec{u}_0) \tag{2.56}$$

uses the sample covariance matrix in (2.38) to obtain

$$\vec{w}(N) = \hat{R}_{xx}^{-1}C(C^H\hat{R}_{xx}^{-1}C)^{-1}\vec{f} \tag{2.57}$$

For the single user case, Reed, et. al. have shown that *if no user signal is present* in the snapshots $\vec{x}(n)$ used to form the sample covariance matrix $\hat{R}_{xx}$, then the weight vector (2.34) typically provides interference suppression to within a few dB of the asymptotic ($N = \infty$) value provided $N \geq 3K$ regardless of the ratio λ_1/λ_K of the eigenvalues of $\hat{R}_{xx}$. The computation required to construct $\hat{R}_{xx}$, and to solve (2.33) is on the order of K^3.

2.5 Beamspace Processing

In many applications, it is useful to decompose the adaptive processor into the cascade of two separate processors as illustrated in Figure 2.12. The first processor is a fixed beamformer. The second processor is an adaptive processor that selects a weight vector $\vec{w}\,'$ so as to minimize the array output power subject to linear constraints on the weight vector $\vec{w}$ of the aggregate processor. The fixed beamformer can be selected to achieve a variety of objectives including: ease of implementation, emphasis of selected spatial sectors relative to others, nulling of emitters in known locations, and reduction of problem dimensionality. We will refer to adaptive processors which have the structure indicated in Figure 2.12 as *beamspace processors.*

2.5.1 Narrowband Case

In the narrowband problem, the beamformer is constructed such that the equivalent weight vector $\vec{w}$ of the aggregate processor is related to the weight vector $\vec{w}'$ of the follow-on adaptive processor by the linear transformation

$$\vec{w} = B\vec{w}' \tag{2.58}$$

where B is a constant $K \times K'$ matrix. Typically,

$$K' \leq K \tag{2.59}$$

We will refer to B as the *beamformer matrix.* If $B^{(i)}$ denotes the i^{th} column of B, then it is clear that (2.58) constrains $\vec{w}$ to be a linear combination of the elementary weight vectors $B^{(1)} \ldots B^{(K')}$.

The output $y(n)$ of the aggregate processor is

$$y(n) = \vec{w}^H \vec{x}(n) = \vec{w}'^H \vec{x}'(n) \tag{2.60}$$

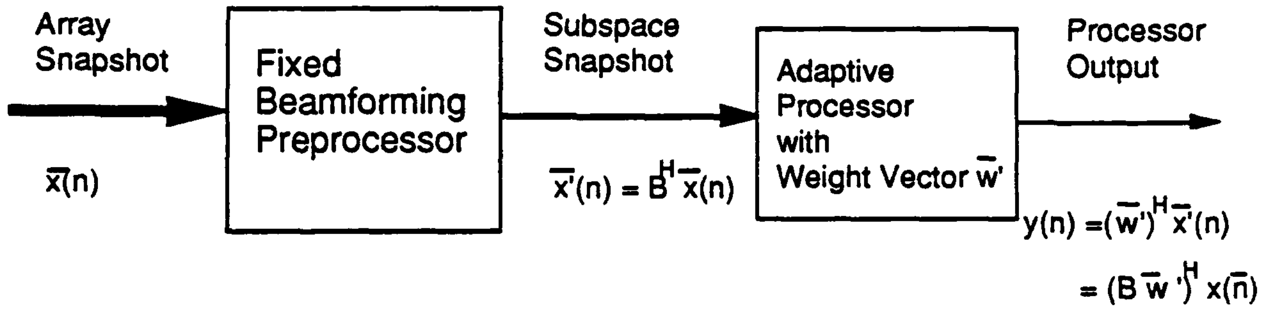

Array Snapshot
$\bar{x}(n)$
Fixed Beamforming Preprocessor
Subspace Snapshot
$\bar{x}'(n) = B^H \bar{x}(n)$
Adaptive Processor with Weight Vector $\bar{w}'$
Processor Output
$y(n) = (\bar{w}')^H \bar{x}'(n)$
$= (B\bar{w}')^H x(\bar{n})$

Figure 2.12: Block Diagram of Beamspace Adaptive Processor

where

$$\vec{x}'(n) = B^H \vec{x}(n) \tag{2.61}$$

Thus, the fixed beamformer can be viewed as transforming the $K \times 1$ snapshot $\vec{x}(n)$ into the $K' \times 1$ snapshot $\vec{x}'(n)$ via (2.61). The i^{th} element of the transformed snapshot is

$$[\vec{x}'(n)]_i = [B^{(i)}]^H \vec{x}(n) \tag{2.62}$$

Therefore, $[\vec{x}'(n)]_i$ corresponds to the output of a fixed weight-and-sum beamformer having the constant weight vector $B^{(i)}$. As a result, the function of the beamformer corresponds to passing $\vec{x}(n)$ through K' fixed weight-and-sum processors having weight vectors $B^{(1)} \cdot\cdot B^{(K')}$, and then taking successive elements of the transformed snapshot $\vec{x}'(n)$ to be the outputs of the successive processors. We will refer to the vectors $\vec{x}'(n)$ and $\vec{w}'$ respectively as the beamspace snapshot vector, and the beamspace weight vector.

The ensemble average power for the aggregate processor is

$$p = E\{|y(n)|^2\} = \vec{w}^H R_{xx} \vec{w} \tag{2.63}$$

where R_{xx} is given by (2.12). Substitution of (2.58) into (2.63) gives

$$p = \vec{w}'^H B^H R_{xx} B \vec{w}' = \vec{w}'^H R'_{xx} \vec{w}' \tag{2.64}$$

where

$$R'_{xx} = E\{\vec{x}'(n) \vec{x}'^H(n)\} = B^H R_{xx} B \tag{2.65}$$

If it is required that the weight vector $\vec{w}$ for the aggregate processor satisfy constraints of the form

$$C^H \vec{w} = \vec{f} \tag{2.66}$$

then substitution of (2.58) into (2.66) shows that the corresponding beamspace constraint on the vector $\vec{w}'$ is

$$C'^H \vec{w}' = \vec{f} \tag{2.67}$$

where

$$C'^H = C^H B \tag{2.68}$$

or

$$C' = B^H C \tag{2.69}$$

The weight vector $\vec{w}'$ that minimizes the power p given by (2.64) subject to the constraint (2.67) is

$$\vec{w}' = R_{xx}'^{-1} C' [C'^H R_{xx}'^{-1} C']^{-1} \vec{f} \tag{2.70}$$

The corresponding weight vector $\vec{w}$ is

$$\vec{w} = B R_{xx}'^{-1} C' [C'^H R_{xx}'^{-1} C']^{-1} \vec{f} \tag{2.71}$$

If B is a non-singular $K \times K$ matrix i.e. $K' = K$, then use of (2.65) and (2.69) in (2.71) shows that

$$\vec{w} = R_{xx}^{-1} C [C^H R_{xx}^{-1} C]^{-1} \vec{f} \tag{2.72}$$

which is identical to the optimum weight vector (2.38) when no beamforming pre-processor is employed. Substitution of (2.72) into (2.64) shows that the corresponding minimum output power p also is identical to (2.39).

Perhaps the best known beamformer matrix for one-dimensional arrays is the Butler Matrix

$$B = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j\phi} & e^{-j2\phi} & & e^{-j(K-1)\phi} \\ 1 & e^{-j2\phi} & e^{-j4\phi} & & e^{-j2(K-1)\phi} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ 1 & e^{-j(K-1)\phi} & e^{-j2(K-1)\phi} & \cdots & e^{-j(K-1)^2\phi} \end{pmatrix} \tag{2.73}$$

where

$$\phi = 2\pi / K \tag{2.74}$$

The matrix (2.73) transforms data from the "element space" of a phased-array with $\lambda/2$ element spacing to the "beam space" of a multi-beam antenna.

2.5.2 Broadband Case

In the broadband problem, the beamformer is constructed such that the $KL \times 1$ *stacked weight vector* $\vec{w}$ for the bank of tapped delay lines is related to a $K' \times 1$ stacked weight vector $\vec{w}'$ for the follow-on adaptive processor by the relationship (2.58), where B is a constant $(KL) \times K'$ matrix. Equation (2.58) constrains the stacked weight vector $\vec{w}$ to be a linear combination of the elementary stacked weight vectors $B^{(1)} \cdot\cdot B^{(K')}$. The transformed snapshot (2.61) corresponds to the result of passing the stacked snapshot $\vec{x}(n)$ through K' fixed transversal filters having the weight vectors $B^{(1)} \cdot\cdot B^{(K')}$, and then taking successive elements of $\vec{x}'(n)$ to be the outputs of successive filters.

If all vectors and matrices in Equations (2.58) - (2.72) are replaced by corresponding vectors and matrices for stacked weight and snapshot vectors, then Equations (2.58) - (2.72) also apply to the broadband case.

2.5.3 Generalized Sidelobe Canceller

Use of an appropriate beamforming pre-processor can simplify many nulling algorithms [7][8].

As an example, consider the power minimization problem defined by Equations (2.63) and (2.66) for the broadband case. Thus, the vectors $\vec{w}$ and $\vec{x}(n)$ denote $KL \times 1$ stacked vectors, and R_{xx} is a $KL \times KL$ matrix. Assume that Q linear constraints are imposed upon $\vec{w}$ so that C in (2.66) is a $KL \times Q$ matrix, and $\vec{f}$ is a $Q \times 1$ vector. [Note that the results can be specialized to the narrowband case simply by taking $L = 1$.]

Consider the $KL \times KL$ beam transformation matrix

$$B = [B1, B2] \tag{2.75}$$

where $B1$ is the $KL \times Q$ matrix

$$B1 = C \tag{2.76}$$

and $B2$ is a $KL \times (KL - Q)$ matrix of the form

$$B2 = [B^{(Q+1)} \cdot\cdot B^{(KL)}] \tag{2.77}$$

Assume that $B2$ is of full rank and also is orthogonal to $B1$; that is,

$$B2^H C = 0 \tag{2.78}$$

Let the beamspace weight vector $\vec{w}'$ be partitioned as follows

$$\vec{w}' = [(\vec{w}1')^T, (\vec{w}2')^T]^T \tag{2.79}$$

where $\vec{w}1'$ is the $Q \times 1$ subvector consisting of the first Q elements of $\vec{w}'$, and $\vec{w}2'$ is the $(KL - Q) \times 1$ subvector consisting of the remaining elements of $\vec{w}'$. That is,

$$\vec{w}1' = [w_1', w_2', \cdot\cdot\, w_Q']^T \tag{2.80}$$

$$\vec{w}2' = [w'_{Q+1} \cdot \cdot w'_{KL}]^T \tag{2.81}$$

Similarly, let the beamspace snapshot $\vec{x}'(n)$ be partitioned as follows

$$\vec{x}'(n) = [(\vec{x}1'(n))^T, (\vec{x}2'(n))^T]^T \tag{2.82}$$

where $\vec{x}1'(n)$ denotes the $Q \times 1$ subvector consisting of the first Q elements of $\vec{x}'(n)$, and $\vec{x}2'(n)$ denotes the $(KL - Q) \times 1$ subvector consisting of the remaining elements of $\vec{x}'(n)$. That is,

$$\vec{x}1'(n) = [[x'(n)]_1 \cdot \cdot [x'(n)]_Q]^T \tag{2.83}$$

$$\vec{x}2'(n) = [[x'(n)]_{Q+1} \cdot \cdot [x'(n)]_{KL}]^T]^T \tag{2.84}$$

where $[\ ]_i$ denotes the i^{th} element of the bracketted vector. Then the output (2.60) of the aggregate adaptive processor shown in Figure 2.12 is

$$y(n) = \vec{w}'^H \vec{x}'(n) = y_1(n) + y_2(n) \tag{2.85}$$

where

$$y_1(n) = (\vec{w}1')^H \vec{x}1'(n) \tag{2.86}$$

and

$$y_2(n) = (\vec{w}2')^H \vec{x}2'(n) \tag{2.87}$$

Thus, the output of the aggregate adaptive processor can be computed as illustrated by the block diagram of Figure 2.13.

Substitution of (2.75) into (2.69) and use of (2.76), (2.78) shows that the beamspace constraint matrix (2.69) is

$$C' = B^H C = \begin{pmatrix} C^H C \\ 0 \end{pmatrix} \tag{2.88}$$

The constraint (2.67) on the beamspace weight vector therefore is

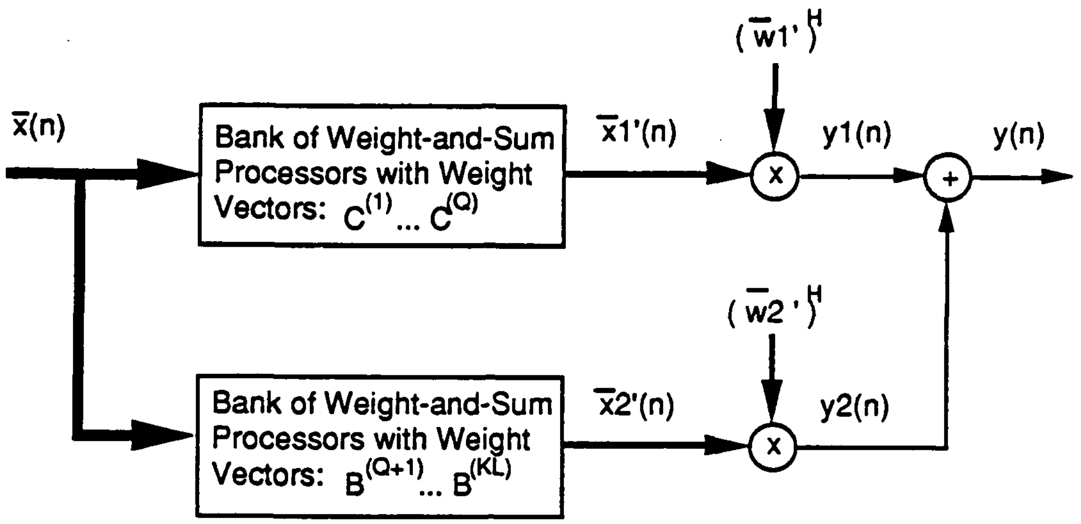
$$C^H C \vec{w}1' = \vec{f} \tag{2.89}$$

so that

$$\vec{w}1' = (C^H C)^{-1} \vec{f} \tag{2.90}$$

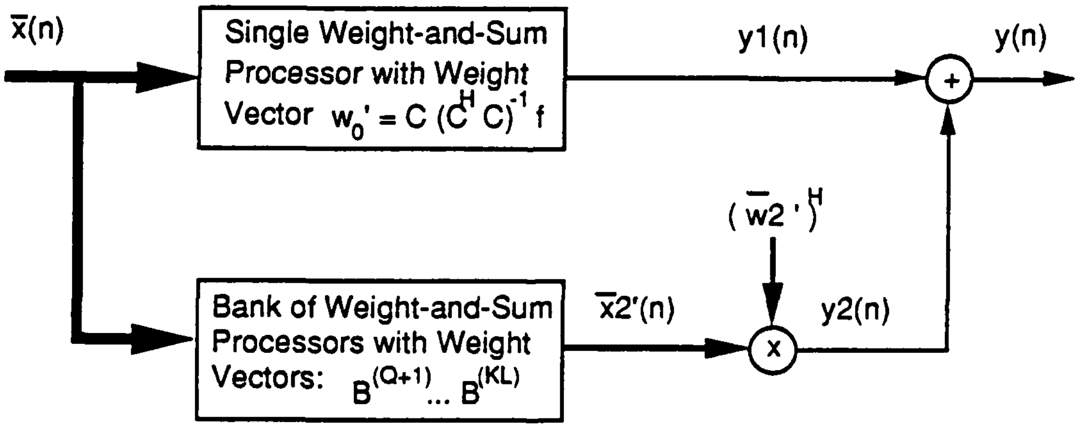
Thus, the constraint (2.66) in the beamspace coordinates simply is that $\vec{w}1'$ equal the constant vector (2.90).

Equation (2.90) fixes the vector $w1'$ in the upper processing path of Figure 2.13, and as a result, the block diagram can be simplified to that shown in Figure 2.14. That is, the



$\bar{x}(n)$
Bank of Weight-and-Sum Processors with Weight Vectors: $C^{(1)} \ldots C^{(Q)}$
$\bar{x}1'(n)$
$(\bar{w}1')^H$
X
y1(n)
+
y(n)
Bank of Weight-and-Sum Processors with Weight Vectors: $B^{(Q+1)} \ldots B^{(KL)}$
$\bar{x}2'(n)$
$(\bar{w}2')^H$
X
y2(n)

Figure 2.13: Block Diagram of Partitioned Beamspace Processor



$\bar{x}(n)$
Single Weight-and-Sum Processor with Weight Vector $w_0' = C\,(C^H C)^{-1} f$
y1(n)
+
y(n)
Bank of Weight-and-Sum Processors with Weight Vectors: $B^{(Q+1)} \ldots B^{(KL)}$
$\bar{x}2'(n)$
$(\bar{w}2')^H$
X
y2(n)

Figure 2.14: Block Diagram of Generalized Sidelobe Canceller

upper processing path can be replaced by a single fixed weight-and-sum processor having the $KL \times 1$ weight vector

$$\vec{w}_0' = C(C^H C)^{-1}\vec{f} \tag{2.91}$$

Thus, *the problem of minimizing (2.68) subject to (2.66) amounts to that of selecting the (reduced-dimension) weight vector $\vec{w}2'$ in the lower processing path of Figure 2.14 so as to minimize the output power.* Owing to its similarity to a sidelobe canceller, the processing structure in Figure 2.14 is known as the Generalized Sidelobe Canceller (GSC)[7][8].

Any convenient adaptive processor now can be used to select the weight vector $w2'$ so as to (approximately) minimize power. For example, a closed-loop Applebaum-Howells processor with KL weights can be employed. In this case, the Q weights $w_1' \cdot \cdot w_Q'$ are clamped to the values specified by (2.90); the remaining weights $w_{Q+1}' \cdot \cdot w_{KL}'$ are allowed to vary and the reference signals for these weights are zero.

The closed-loop LMS algorithm also can be employed. In this case, the weight vector is updated according to the rule

$$\vec{w}2'(n+1) = \vec{w}2'(n) - \alpha\vec{x}2'(n) \cdot y'^*(n) \tag{2.92}$$

where α is a small positive constant. This implementation of LMS is noteworthy in that it does not require introduction of a pilot signal.

Alternately, an open-loop SMI processor can be utilized. In this case, an estimate $\hat{R}_{xx}'$ of R_{xx}' is computed from the expression

$$\hat{R}_{xx}' = \frac{1}{M}\sum_{n=1}^{N} \vec{x}'(n)\vec{x}'^H(n) \tag{2.93}$$

and the result, together with (2.71), is used in the expression to obtain $\vec{w}'$.

2.5.4 Sub-space Techniques

While the apparent dimension of a narrowband nulling problem is the number K of sensors, the true dimensionality is $J+1$ where J denotes the number of jammers. Thus, in applications for which $K \gg (J+1)$ substantial performance and computation benefits can be achieved by appropriately reducing the dimension of the problem. Techniques which exploit such a reduction in dimension are called *sub-space techniques.*

The linear transformation (2.58) can provide an effective method for reducing dimensionality. The transformation reduces the number of unknown weights from K to K'. The objective in using the transformation is to not compromise performance in the process. In principle, if the matrix B satisfies the following conditions, then the beamspace weight set has a number of degrees of freedom sufficient to null J jammers

1. The columns of B are linearly independent.

2. $K' \geq (J+1)$

Although a weight set satisfying these conditions may possess the number of degrees of freedom sufficient to null J jammers, it will not be possible to accurately calculate the weight vector if the jammer signals are not strongly present in the beamspace snapshot $\vec{x}'$. Thus, it is desirable that the matrix B also have the property that

3. The columns of B approximately span the vector space defined by the unit-arrival vectors $\vec{u}_1 \cdot \cdot \vec{u}_J$ for the jammers, and also those for the users.

One method of constructing a suitable matrix B is to 1) perform a coarse direction-of-arrival analysis [e.g., by means of the Fast-Fourier Transform for uniform 1D and 2D arrays]; 2) generously estimate the number K' of active emitters and their approximate directions (including user directions); and, 3) take the columns of B to be unit-arrival vectors for the estimated emitter directions.

Provided the beams are well chosen, sub-space techniques can produce both improved performance and reduced computation compared to techniques that employ a weight vector with a maximum number K of degrees of freedom. Our Phase I work[11] demonstrated that sub-space techniques can be used to accelerate convergence of variants of the SMI algorithm, while reducing the required computation from $O(K^3)$ to $O(J^2K)$. More recently, Gabriel[12] has shown that a sub-space implementation of GSC which retains only J beams in the neighborhood the jammers in the lower processing path of Figure 2.14, and which utilizes the LMS algorithm for weight selection, provides more stable convergence than a full K-degree of freedom implementation of LMS, while similarly reducing computation.

Bibliography

[1] A. Oppenheim and R. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[2] P.W. Howells, "Exploration in Fixed and Adaptive Resolution at GE and SURC", *IEEE Trans. Antennas and Propagat.*, Vol. AP-24, pp. 575-584, Sept. 1976.

[3] S. P. Applebaum, "Adaptive Arrays", *IEEE Trans. Antennas and Propagat.*, Vol. AP-24, pp. 585-598, Sept. 1976.

[4] B. Widrow, P. Mantey, L. Griffiths, B. Goode, "Adaptive Antenna Systems", *Proc. IEEE,* Vol. 55, pp. 2143-2158, 1967.

[5] L. Griffiths, "A Simple Adaptive Algorithm for Real-Time Processing in Antenna Arrays", *Proc. IEEE,* Vol. 57, pp. 1696-1704, 1969.

[6] O. Frost, "An Algorithm for Linearly Constrained Adaptive Array Processing", *Proc. IEEE,* Vol. 60, pp. 661-675, 1971.

[7] L.J. Griffiths and C.W.Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming", *IEEE Trans. Antennas and Propagat.*, Vol. AP-30, No. 1, pp. 27-34, January 1982.

[8] K. Buckley, "Spatial/Spectral Filtering with Linearly Constrained Minimum Variance Beamformers", *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-35, No. 3, pp. 249-266, Mar. 1987.

[9] I.S. Reed, J.D. Mallet and L.E. Brennan, "Rapid Convergence Rate in Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-10, No. 6, pp. 853-863, Nov. 1974.

[10] K. Takao and K. Komiyama, "An Adaptive Antenna for Rejection of Wideband Interference", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-16, pp.452-459, July 1980.

[11] "Advanced Nulling Techniques: Phase I Final Report", Atlantic Aerospace Electronics Corporation (Formerly Pollard Road Inc.), AF Contract Number F04701-85-C-0078, Mar. 1986.

[12] W.F. Gabriel, "Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems", *IEEE Trans. Antennas and Propagat.*, Vol. AP-34, No. 3, pp. 291-300, Mar. 1986.

Chapter 3

Fast Open-Loop Algorithms

Contents

List of Figures

3.1 Introduction

Given the advent of frequency-hopped communications signals and smart jammers, the modern electromagnetic environment has become highly *non-stationary.* By contrast, traditional closed-loop nulling algorithms were developed for slowly-varying environments. Thus, it is reasonable to expect that traditional algorithms will be of reduced utility in highly non-stationary environments.

The purpose of this chapter is to explore the development of rapidly converging nulling algorithms that are applicable to highly non-stationary environments. We take the Sample Matrix Inversion (SMI) algorithm as our starting point (see Section 2.4.3). SMI assumes the weight vector for the adaptive processor to be the solution of the approximate Weiner-Hopf equation,

$$\vec{w}(N) = \mu \hat{R}_{xx}^{-1} \vec{u}_0 \tag{3.1}$$

or more generally takes the weight vector, for the multiple constraint problem, given by

$$\vec{w}(N) = \hat{R}_{xx}^{-1} C (C^H \hat{R}_{xx}^{-1} C)^{-1} \vec{f} \tag{3.2}$$

It is well known that 1) the *SMI weight vector rapidly suppresses interference provided that the user signal is not present in the snapshots used to construct the sample covariance matrix* $\hat{R}_{xx}$, and 2) the *transient performance of SMI degrades dramatically as the user signal is introduced into the snapshots.* Unfortunately, the user signal typically is present in snapshots for communications problems. Thus, *treatment of the user signal emerges as the fundamental problem in developing rapidly convergent nulling algorithms.*

We present a new class of nulling algorithms that largely overcomes the problems introduced by the user signal. The essence of the algorithms is to use a priori user directional information to spatially "filter out" the user signal from the data prior to calculating the adaptive nulling weights. The algorithms are applicable to one, two, and three dimensional convolu .onal arrays. However, the algorithms have not been extended to completely general arrays. It is hoped that the positive results obtained for convolutional arrays will motivate development of similar algorithms for arbitrary arrays.

This chapter is organized as follows. Section 3.2 documents the adverse impact of user signal(s) upon the transient response of SMI and related algorithms. Section 3.3 describes the new class of algorithms that accelerate the convergence rate by filtering out the user signal prior to weight calculation. Section 3.4 presents examples illustrating the comparative performance of the algorithms for one-dimensional arrays. Section 3.5 presents examples for two-dimensional arrays.

3.2 Impact of the User Signal on Transient Response

The purpose of this section is to clarify the impact of the user signal on the transient performance of an adaptive array. Section 3.2.1 reviews the status of the problem in the literature. Section 3.2.2 shows that the *solution of the Weiner-Hopf equation is independent of the presence/absence of the user signal.* This result also shows that the solutions of the approximate Weiner-Hopf equation approach the same limit, as the number of snapshots N approaches infinity, regardless of the level of the user signal. Sections 3.2.3 and 3.2.4 demonstrate by examples that the situation is quite different for small values of N.

Specifically, it is shown that:

1. The interference suppression provided by the weight vector which solves the approximate Weiner-Hopf equation is strongly dependent upon the level of the user signal in the snapshots used to construct $\hat{R}_{xx}$,
2. The degradation in interference suppression is not a result of the spread in the eigenvalues of $\hat{R}_{xx}$.

Section 3.2.5 shows that, for finite N, the adaptive processor typically mistakes the user signal for a mainbeam jammer, and places a null near the user. Section 3.2.6 shows by example that by appropriately "dividing out" this null, a weight vector is produced with greatly improved properties.

3.2.1 Prior Work on Speed of Convergence

The purpose of this section is to summarize prior work on the convergence speed of adaptive processors.

Closed-Loop Algorithms

Analysis of the transient response of closed-loop adaptive processors has been somewhat limited. The analyses typically are based on the eigenvalues of the steady-state covariance matrix R_{xx} rather than on the properties of the time-averaged covariance matrix $\hat{R}_{xx}$[1][2]. The results show that the expected value of the weight vector $E\{\vec{w}(N)\}$ converges exponentially to the (optimum) weight vector that solves the Weiner-Hopf equation, and that the exponential time constants are function of the eigenvalues of R_{xx} and implementation parameters [eg. step size, filter bandwidths, etc.]. No results are available to show that the solution of the approximate Weiner-Hopf equation (3.1) is close to $E\{\vec{w}(N)\}$ for finite N. Thus, the results obtained for $E\{\vec{w}(N)\}$ provide little insight into the behavior of the ratio $\hat{p}(N)/\hat{p}(\infty)$ where $\hat{p}(N)$ denotes the array output power.

Indeed, the performance properties of closed-loop processors appear to have been established almost entirely on the basis of simulations. Typically, the simulations have shown that convergence times are scenario-dependent and can be quite large in unfavorable scenarios[3].

With regard to the impact of the user signal upon the speed of convergence, few, if any, published results are available. Widrow et.al. have studied steady-state cancellation effects caused by broadband user and interfering signals in adaptive array processors[4] The so-called *Duvall Beamformer* was shown to significantly reduce adverse cancellation effects[4]. The beamformer consists of a pre-processor followed by a Frost Beamformer[5]. The pre-processor eliminates a broadside user signal by differencing the outputs of adjacent sensors of a uniform linear array; the Frost Beamformer then processes the differenced signals.

Open Loop Algorithms

More comprehensive results are available for open-loop algorithms[8]. The results show that ***algorithms like SMI converge rapidly regardless of scenario, provided the user signal is absent from the snapshots used to compute the sample covariance matrix*** $\hat{R}_{xx}$. Specifically Reed et.al.[3] show that the SMI weight vector produces noise-plus-interference suppression that is within a few dB of the asymptotic value ($N = \infty$) provided that

$$N \geq 3K \tag{3.3}$$

Related work by Boroson[6] and Miller[7][8] has shown that ***the speed of convergence of SMI degrades dramatically as the user signal is introduced into the snapshots used to calculate*** $\hat{R}_{xx}$. While these results are useful in quantifying the degradation, they do not identify the root cause, nor do they suggest improvements.

In our previous work[9] we noted that the result (3.3) can be sharpened to

$$N \geq 3(J+1) \tag{3.4}$$

for applications in which the interference consists of J spatially discrete narrowband jammers. We also have shown via simulations that for the case of a uniform linear array, the degradation in convergence rate due to the user signal largely can be eliminated by preprocessing snapshots in the same manner as the Duvall Beamformer. Specifically, signals from adjacent sensors are appropriately phased and differenced to remove the user signal; then the differenced signals are processed by a suitable adaptive processor to determine nulling weights.

3.2.2 Asymptotic Value of the Weight Vector Solution

The purpose of this section is to show that the solution $\vec{w}$ of the exact Weiner equation

$$\vec{w} = \mu R_{xx}^{-1} \vec{u}_0 \tag{3.5}$$

is independent of the level of the user signal. Since $\hat{R}_{xx} \rightarrow R_{xx}$ as $N \rightarrow \infty$, it follows that the limiting solution of the approximate Weiner-Hopf equation (3.1) is (3.5), and therefore also is independent of the user signal.

If a single user signal is present in the snapshot data used to construct the covariance matrix, then the expected value of the covariance matrix can be expressed as follows:

$$R_{xx} = R_{nn} + \Pi_0 \vec{u}_0 \vec{u}_0^H \tag{3.6}$$

where $\vec{u}_0$ denotes the unit-arrival vector for the user signal, $K\Pi_0$ denotes the power in the user signal at a sensor, and R_{nn} denotes the covariance matrix for the interference and noise environment (with the user signal absent). It follows from the matrix identity

$$(A + \vec{x}\vec{x}^H)^{-1} = A^{-1} - A^{-1}\vec{x}\vec{x}^H A^{-1}/(1 + \vec{x}^H A^{-1} \vec{x}) \tag{3.7}$$

that

$$R_{xx}^{-1} \vec{u}_0 = (R_{nn} + \Pi_0 \vec{u}_0 \vec{u}_0^H)^{-1} \vec{u}_0 = \alpha R_{nn}^{-1} \vec{u}_0 \tag{3.8}$$

where

$$\alpha = [1 - \Pi_0(\vec{u}_0^H R_{nn}^{-1} \vec{u}_0)]/[1 + \Pi_0(\vec{u}_0^H R_{nn}^{-1} \vec{u}_0)] \tag{3.9}$$

Use of (3.8) in the exact Weiner-Hopf equation (3.5) gives

$$\vec{w} = \mu R_{nn}^{-1} \vec{u}_0 \tag{3.10}$$

where

$$\mu = 1/(\vec{u}_0^H R_{nn}^{-1} \vec{u}_0) \tag{3.11}$$

Thus the solution of the exact Weiner Hopf equation (3.5) is the same whether or not the user signal is present.

The solutions of the approximate Weiner-Hopf equation (3.1) approach the same asymptotic value as $N \to \infty$ regardless of the present/absence or level of the user signal.

3.2.3 Transient Behavior of SMI

Nonwithstanding the fact that the solutions of (3.1) approach the same limit as $N \to \infty$ regardless of the level of the user signal, the transient behavior of the solution vector and also of the associated interference suppression depend strongly upon the level of the user signal. The following example illustrates this point.

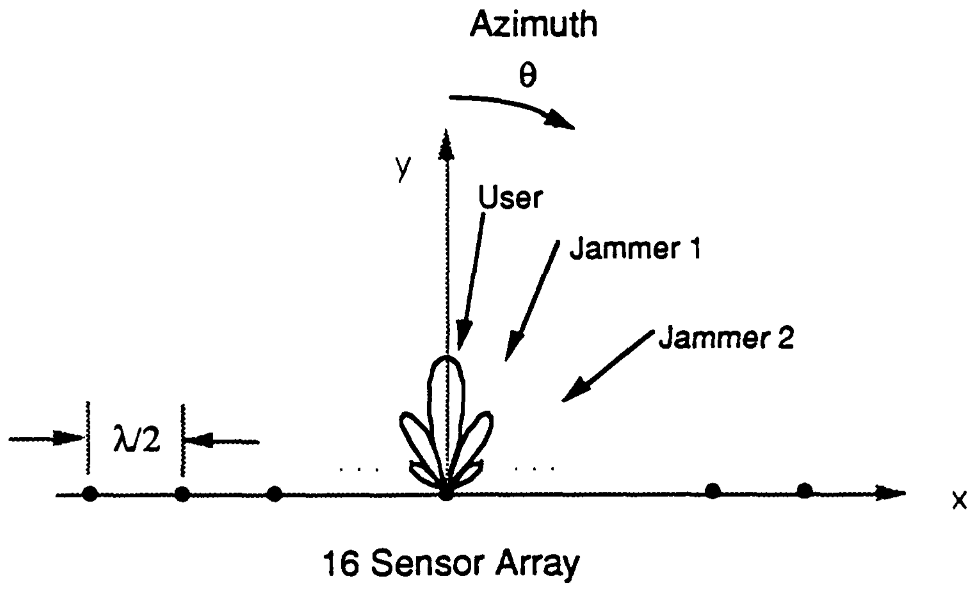
Example 3.1

Consider a linear receiving array consisting of 16 omnidirectional sensors with half wavelength $\lambda/2$ spacing as illustrated in Figure 3.1.

Assume that a single user is located at an off-broadside angle of $\theta = 5°$, and that two jammers transmit from angles of $\theta = 15°$ and $\theta = 50°$. Figure 3.2 illustrates the positions of the jammers with respect to the antenna pattern for a uniform-illumination weight set steered at the user. The jammers are located near the peaks of sidelobes of the quiescent pattern. Figure 3.3 depicts the antenna pattern for the asymptotic ($N = \infty$) weight vector. Initially assume that when the user is "on" it transmits a constant-envelope waveform that produces a sensor signal-to-noise ratio (SNR) of 30 dB (0 dB signal power, -30 dB noise power). Also assume that the jammers continuously transmit uncorrelated constant envelope signals each of which produces a sensor jammer-to-noise ratio (JNR) of 30 dB (0 dB jammer power, -30 dB noise power).

Figure 3.4 contrasts the interference suppression produced by SMI for the cases of "user present" and "user absent". The horizontal axis denotes the number N of snapshots used to construct the sample covariance matrix $\hat{R}_{xx}$. The vertical axis depicts the array output signal-to-interference-plus-noise ratio (SINR) averaged over 10 Monte Carlo trials for the weight vector (3.1). Note that for the case of no user, the SMI weight vector produces interference suppression of 25 dB relative to a uniform illumination weight vector after only 6 ($= 3J$) snapshots, and provides suppression to within a few dB of the asymptotic value of SINR = 41.8 dB.

By contrast, for the case of the user present, the SMI weight vector provides no useful



User: $\theta = 15^\circ$; SNR = 30 dB

Jammers: $\theta = 15^\circ, 50^\circ$; JNR =30 dB

Figure 3.1: Geometry for Linear Array Examples

Figure 3.2: Antenna Pattern for Uniform-Illumination Weight Set Which Steers Toward User

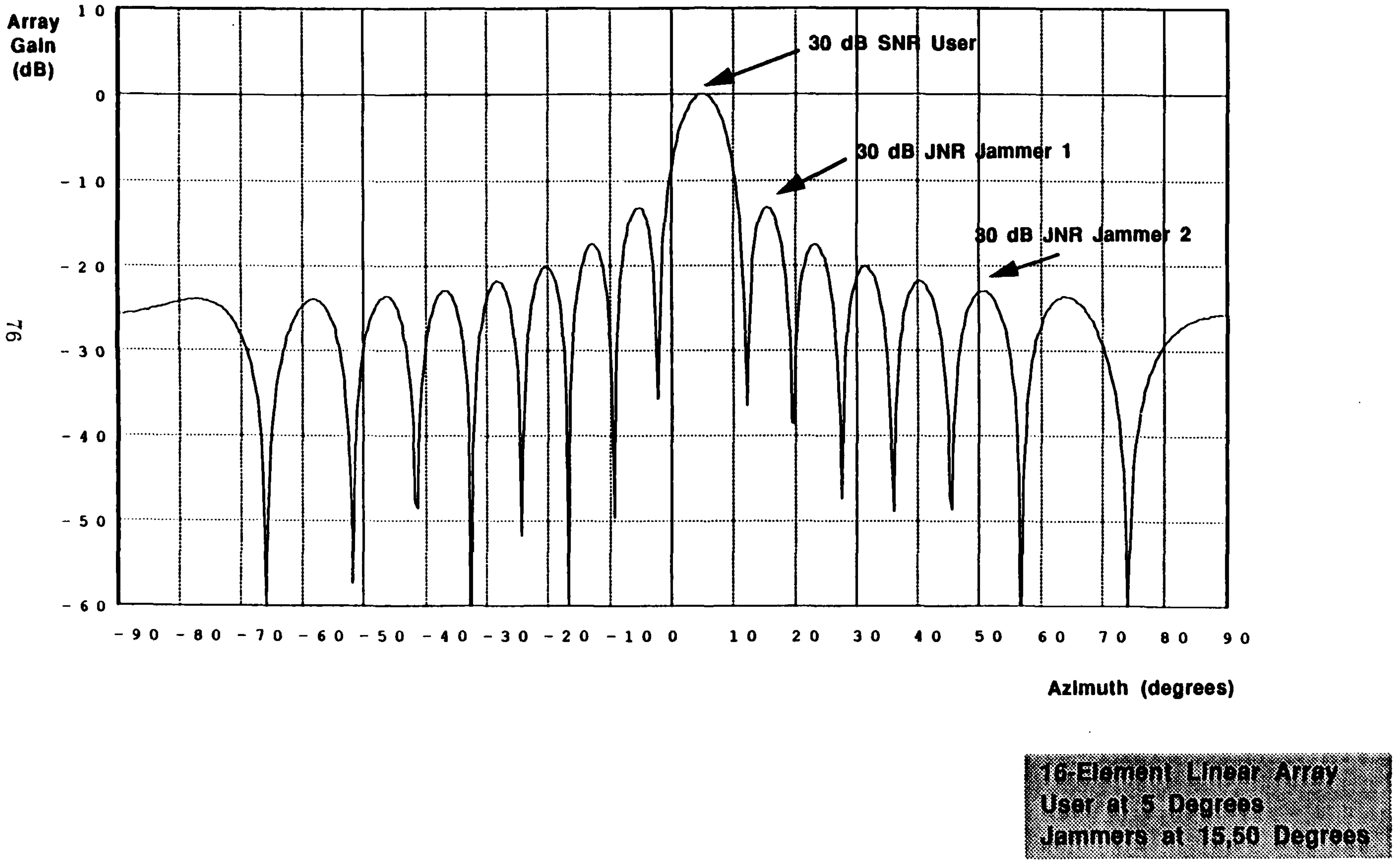


Figure 3.3: Antenna Pattern for the Asymptotic Weight Vector

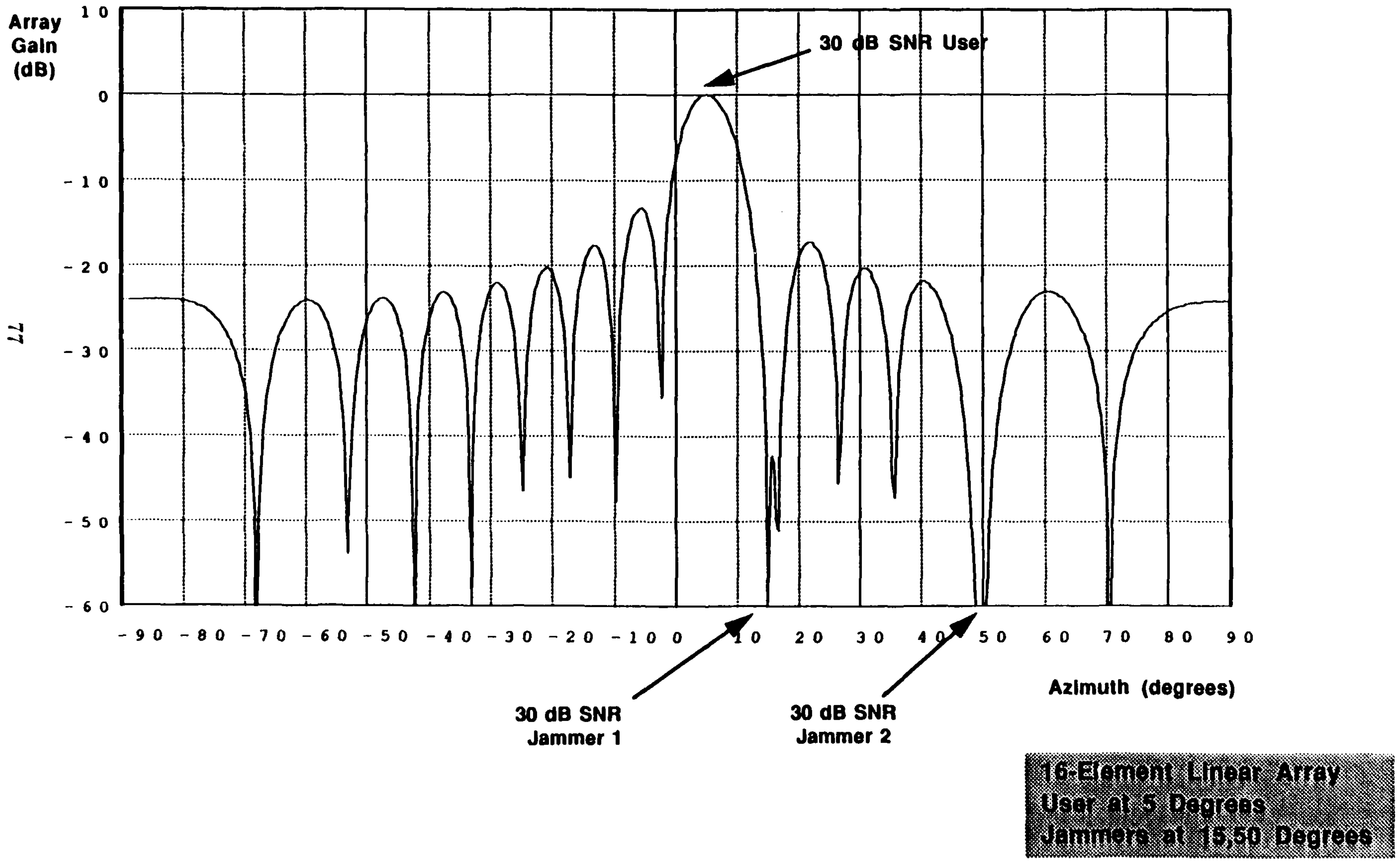
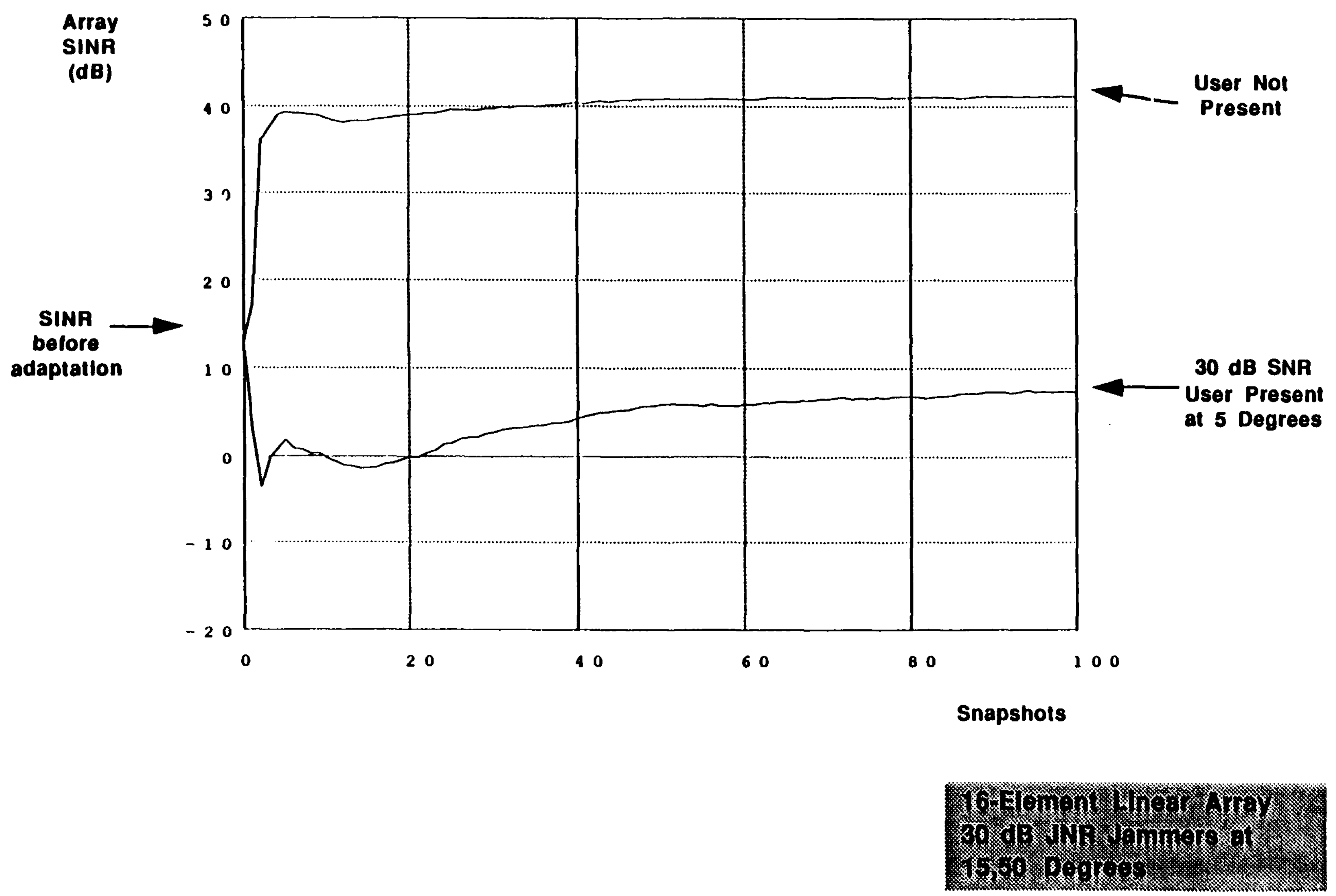


Figure 3.4: Effect of User Presence on Array SINR



suppression even after 100 snapshots.

Performance degradation in adaptive arrays often is explained in terms of the eigenvalue spread of the matrix $\hat{R}_{xx}$ and R_{xx}. Table 3.1 summarizes the eigenvalues, $\lambda_1, \lambda_2, \cdots \lambda_{16}$ of $\hat{R}_{xx}$ and R_{xx} for both the user-present and user-absent cases.

	User Absent		User Present	
	R_{xx}	$\hat{R}_{xx}$	R_{xx}	$\hat{R}_{xx}$
λ_1	1.016	1.180	1.052	1.247
λ_2	0.984	0.825	0.985	1.015
λ_3	6.250e-5	1.028e-4	0.963	0.742
λ_4	6.250e-5	9.333e-5	6.250e-5	9.848e-5
.	.	.	.	.
λ_{16}	6.250e-5	2.537e-5	6.250e-5	2.621e-5

Table 3.1: Eigenvalues for Steady-State and 100 Snapshot Covariance Matrices Without and With User Signal Present

A total of $N = 100$ snapshots were used to calculate $\hat{R}_{xx}$. Since $\hat{R}_{xx} \rightarrow R_{xx}$ as $N \rightarrow \infty$, the matrix R_{xx} corresponds to an infinite snapshot matrix. It is evident from Table 3.1 that there is no significant difference between the spread of the eigenvalues of $\hat{R}_{xx}$ and those of R_{xx}. On the other hand, the solution of (3.1) provides poor suppression, while the solution of (3.5) provides good suppression. Thus, ***the degradation in suppression due to the user signal presence is an effect that is not traceable to the spread in eigenvalues.*** Instead the degradation is due to a different mechanism (sec Section 3.2.5).

Figure 3.5 further depicts the performance of the adaptive array as the power level of the user signal is stepped through a range of values, and the jammers maintain their former characteristics. The figure depicts the interference-plus-noise output power of the array for the weight vector (3.1) versus the number N of snapshots used to calculate $\vec{w}(N)$. For signal levels below the noise floor (i.e. 0 dB SNR) the array adapts rapidly and provides useful interference suppression. However, for signal levels above the noise floor suppression degrades almost dB for dB with increases in the user signal level.

Figure 3.6 depicts the array interference-plus-noise output power over an interval corresponding to 1000 snapshots. The curves clearly are tending toward the indicated asymptotic

Figure 3.5: A⁊aptive Array Interference-Plus-Noise (I+N) For Different Levels of User Signal

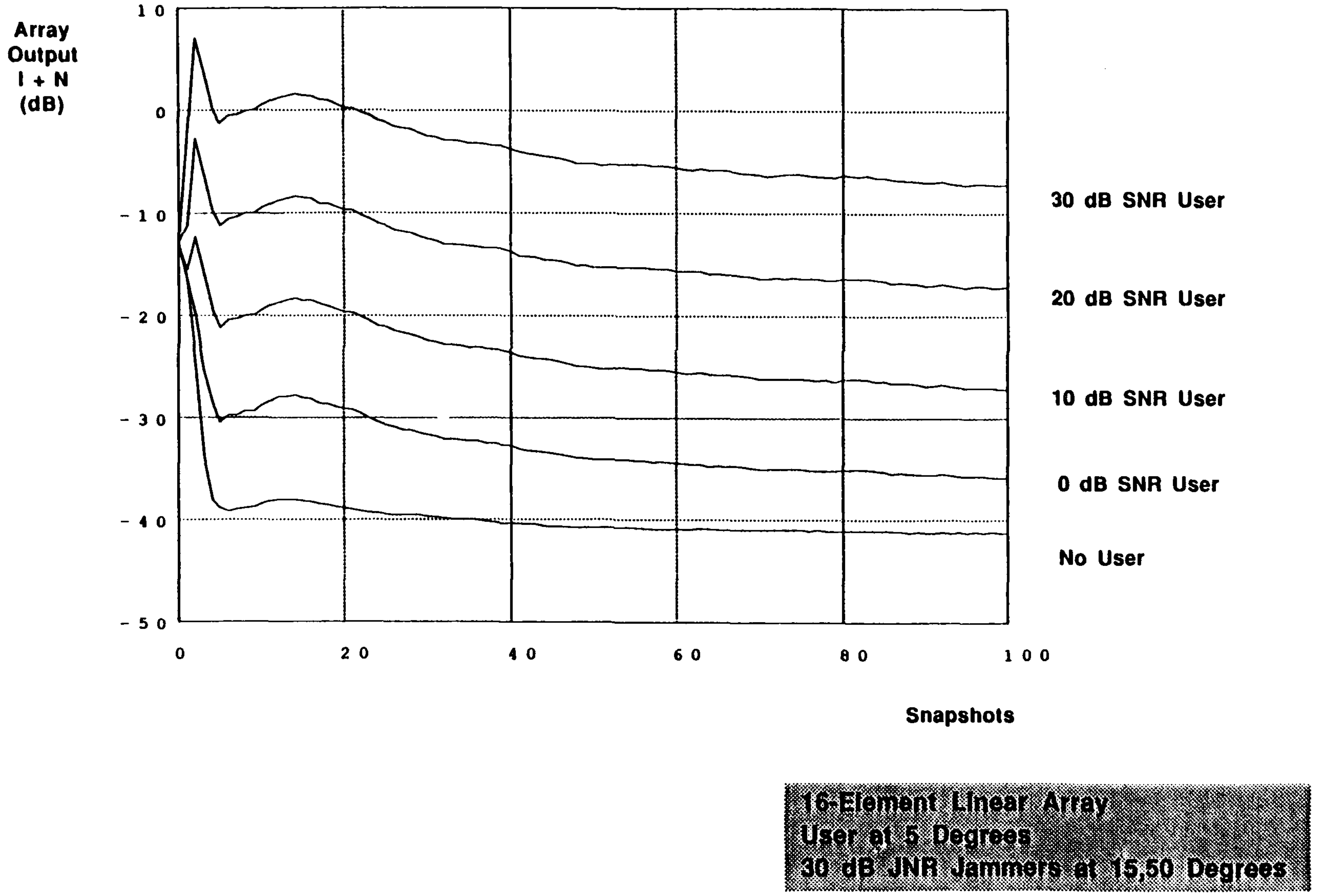
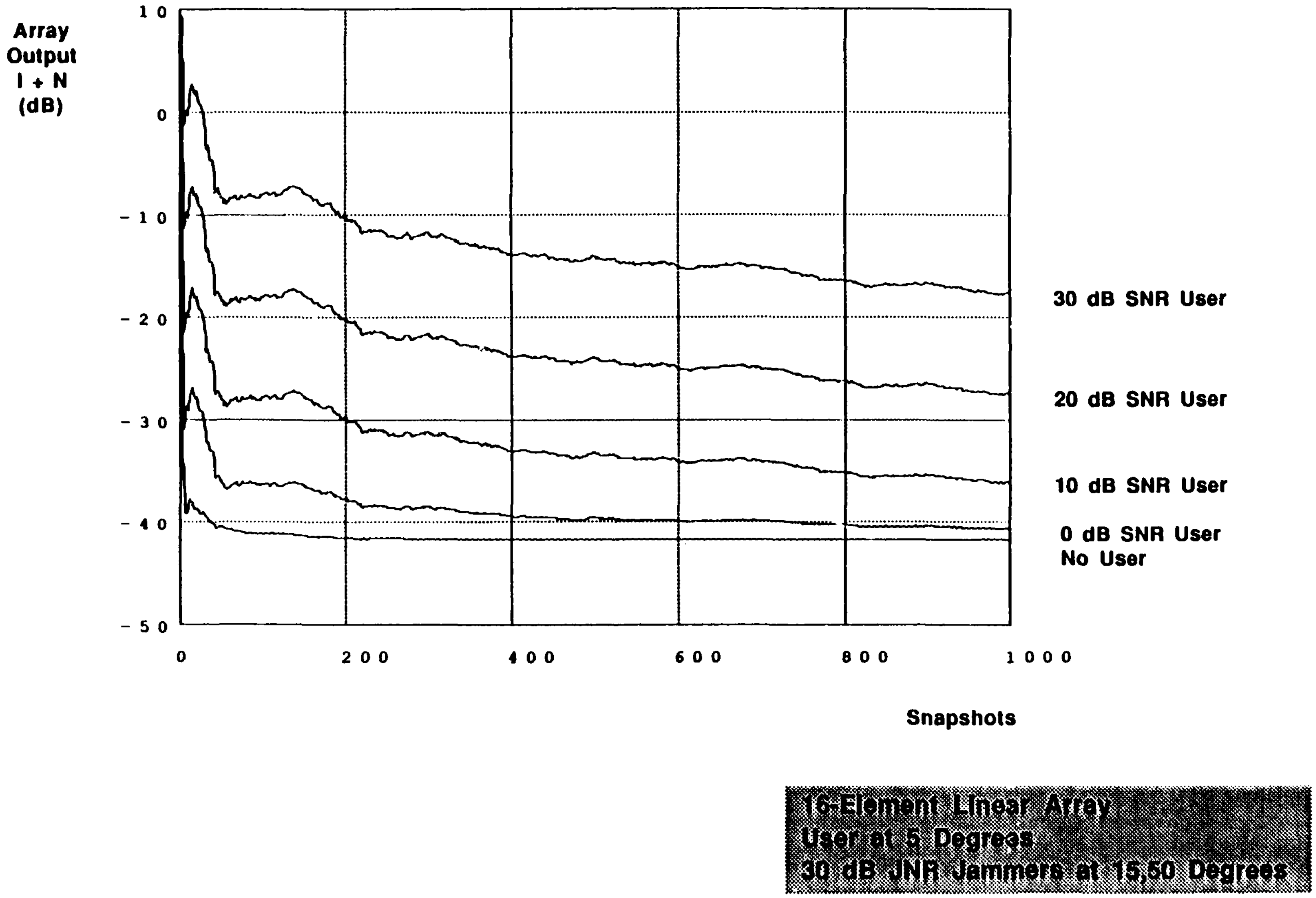


Figure 3.6: Adaptive Array Interference-Plus-Noise (I+N) For Different Levels of User Signal



value as the weight vectors approach the (common) asymptotic value.

End of Example 3.1

3.2.4 Transient Behavior of the Generalized Sidelobe Canceller (GSC)

For the case of a single gain constraint to a user, the GSC structure [10][11] takes the form illustrated in Figure 2.14. The upper signal path consists of a fixed weight-and-sum processor which steers a beam at the user. Thus, the upper path passes both the user and jammers. The lower signal path of Figure 2.14 consists of $K - 1$ fixed weight-and-sum processors with variable gain controls. The weight vectors for these processors are orthogonal to that for the upper-path processor; therefore the lower-path processors null the user and pass much of the jammer power. As indicated in Section 2.5.3, the gain controls in the lower path are adapted so as to minimize the array output power $E\{|y|^2\}$.

Given that the GSC configuration nulls the user in the lower path of Figure 2.14, it might be thought that the use of SMI to select the variable weights in the lower path of the GSC circumvents the convergence problems associated with conventional SMI when the user signal is present. However, as discussed in Section 2.5.3, the GSC weight set is simply a linear transformation of the conventional SMI weight set. Thus, application of SMI in the new coordinate system provides no benefit with regard to accelerating the convergence behavior. The following example illustrates this point.

Example 3.2

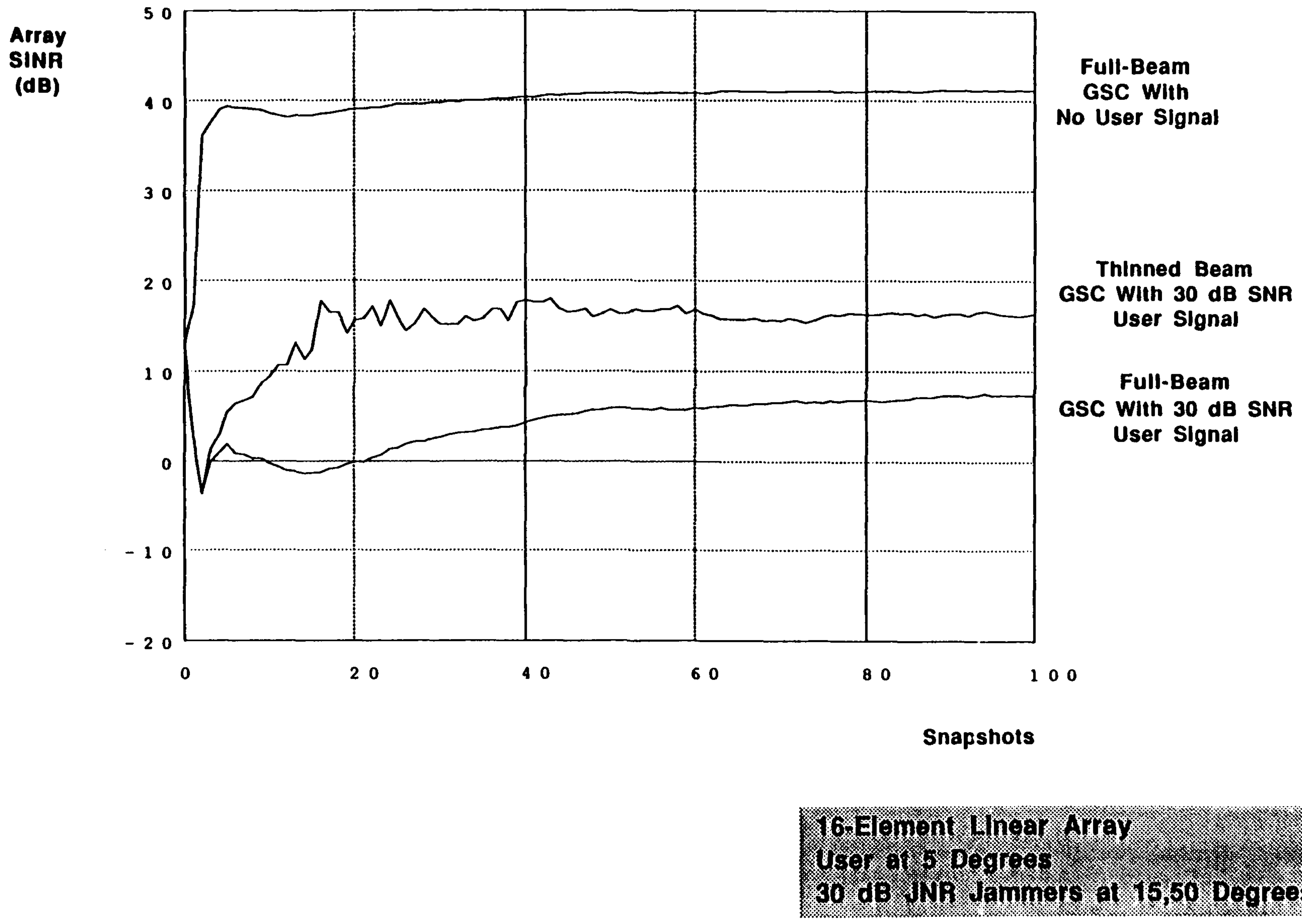
Figure 3.7 depicts the results of using SMI to compute the weights for the GSC configuration shown in Figure 2.14 for the following two cases:

1. The lower path utilizes a complete set of $K - 1$ linearly-independent weight-and-sum processors.
2. The lower path utilizes a *thinned set* of two processors each of which steers a beam at a jammer.

The use of a thinned processor set has been shown by Gabriel[12] to provide benefits in problems where the user signal is absent.

The curves of Figure 3.7 depict the output SINR versus the number of snapshots used to compute the weight vector for cases 1 and 2. The upper and lower curves show that

Figure 3.7: Performance of GSC With and Without User Present



introduction of the user signal severely degrades the short-term suppression afforded by the array. In fact, comparison of Figure 3.7 with Figure 3.4 shows that the GSC curves with the full beam set are identical to the SMI curves as was discussed earlier.

The middle curve of Figure 3.7 shows that a modest improvement in suppression is achieved with the user signal present if the thinned set of weight-and-sum processors is used in lower path of Figure 2.14 rather than the full beam set. However, the overall performance is still 20-25 dB poorer than that of GSC or SMI without the user present. Performance is similarly degraded relative to that of the modified SMI technique illustrated in Example 3.3

End of Example 3.2

3.2.5 Adaptive Processor Typically Nulls the User Signal

To gain some insight into the nature of the transient response of SMI and related algorithms, consider the solution of the approximate Weiner-Hopf equation (3.1), and also the solution of the exact Weiner-Hopf equation (3.5). For large signal-to-noise ratios the coefficient matrices $\hat{R}_{xx}$ and R_{xx} can be expressed in terms of their eigenvalues and eigenvectors as follows

$$\hat{R}_{xx} = \left(\hat{E}_s\ \hat{E}_n\right) \begin{pmatrix} \hat{\Lambda}_s & 0 \\ 0 & \hat{\Lambda}_n \end{pmatrix} \begin{pmatrix} \hat{E}_s^H \\ \hat{E}_n^H \end{pmatrix} \tag{3.12}$$

and

$$R_{xx} = \left(E_s\ E_n\right) \begin{pmatrix} \Lambda_s & 0 \\ 0 & \Lambda_n \end{pmatrix} \begin{pmatrix} E_s^H \\ E_n^H \end{pmatrix} \tag{3.13}$$

where $\hat{\Lambda}_s$ (Λ_s) is a diagonal submatrix, the diagonal elements of which are the $J+1$ largest eigenvalues of $\hat{R}_{xx}$ (R_{xx}), $\hat{E}_s$ (E_s) is a submatrix consisting of corresponding "signal-space" eigenvectors, $\hat{\Lambda}_n$ (Λ_n) is a diagonal submatrix the diagonal elements of which are the $K-(J+1)$ "small" eigenvalues of $\hat{R}_{xx}$ (R_{xx}), and $\hat{E}_n$ (E_n) is a submatrix consisting of the corresponding "noise sub-space" eigenvectors.

The inverse of $\hat{R}_{xx}$ is given by

$$\hat{R}_{xx}^{-1} = \hat{E}_s\hat{\Lambda}_s^{-1}\hat{E}_s^H + \hat{E}_n\hat{\Lambda}_n^{-1}\hat{E}_n^H \tag{3.14}$$

so that (3.1) can be written as

$$\vec{w} = \mu\hat{R}_{xx}^{-1}\vec{u}_0 = \mu\hat{E}_s\hat{\Lambda}_s^{-1}\hat{E}_s^H\vec{u}_0 + \mu\hat{E}_n\hat{\Lambda}_n^{-1}\hat{E}_n^H\vec{u}_0 \tag{3.15}$$

For high JNR, high SNR (when the user is present), and moderate N, the weight vector is

$$\vec{w} \approx \mu \hat{E}_n \hat{\Lambda}_n^{-1} \hat{E}_n^H \vec{u}_0 \tag{3.16}$$

Thus, the weight vector (3.16) tends to be a linear combination of the noise subspace vectors. Since the noise subspace vectors for finite N are approximately, but not exactly, orthogonal to the unit-arrival vectors for the active *emitters* (i.e. the columns of E_s), the weight vector also is approximately orthogonal to the arrival vectors corresponding to the active emitters. Thus, if no user signal is present in the data used to construct $\hat{R}_{xx}$, then the weight vector (3.16) typically suppresses the jammers but not the user.

On the other hand, if the user signal is present in the snapshots used to construct $\hat{R}_{xx}$, then the weight vector (3.16) tends to suppress (or null) *both the user and the jammers.* Subsequent normalization to provide unit gain in the direction of the user proportionally increases the gains to the jammers, and also greatly magnifies the thermal noise power. As a result, significantly increased jammer power and thermal power is output by the array. Indeed, *the stronger the user signal, the sharper the null on the user, the larger the normalizing constant required to provide unit gain to the user, and the larger the output interference and thermal noise power.*

As $N \to \infty$, the weight vector (3.16) approaches the asymptotic value (3.10) regardless of the presence/absence of the user signal. Thus, whereas the transient value of the weight vector approximates the linear combination (3.16) of transient noise subspace eigenvectors, the asymptotic value (3.10) is a linear combination of the asymptotic signal-space eigenvectors and the vector $\vec{u}_0$, since

$$\begin{aligned} R_{xx}^{-1} &= E_s \Lambda_s^{-1} E_s^H + E_n \Lambda_n^{-1} E_n^H \\ &= E_s \Lambda_s^{-1} E_s^H + \sigma^{-2} E_n E_n^H \\ &= E_s \Lambda_s^{-1} E^H + \sigma^{-2}(I - E_s E_s^H) \\ &= E_s(\Lambda_s^{-1} - \sigma^{-2} I) E_s^H + \sigma^{-2} I \end{aligned} \tag{3.17}$$

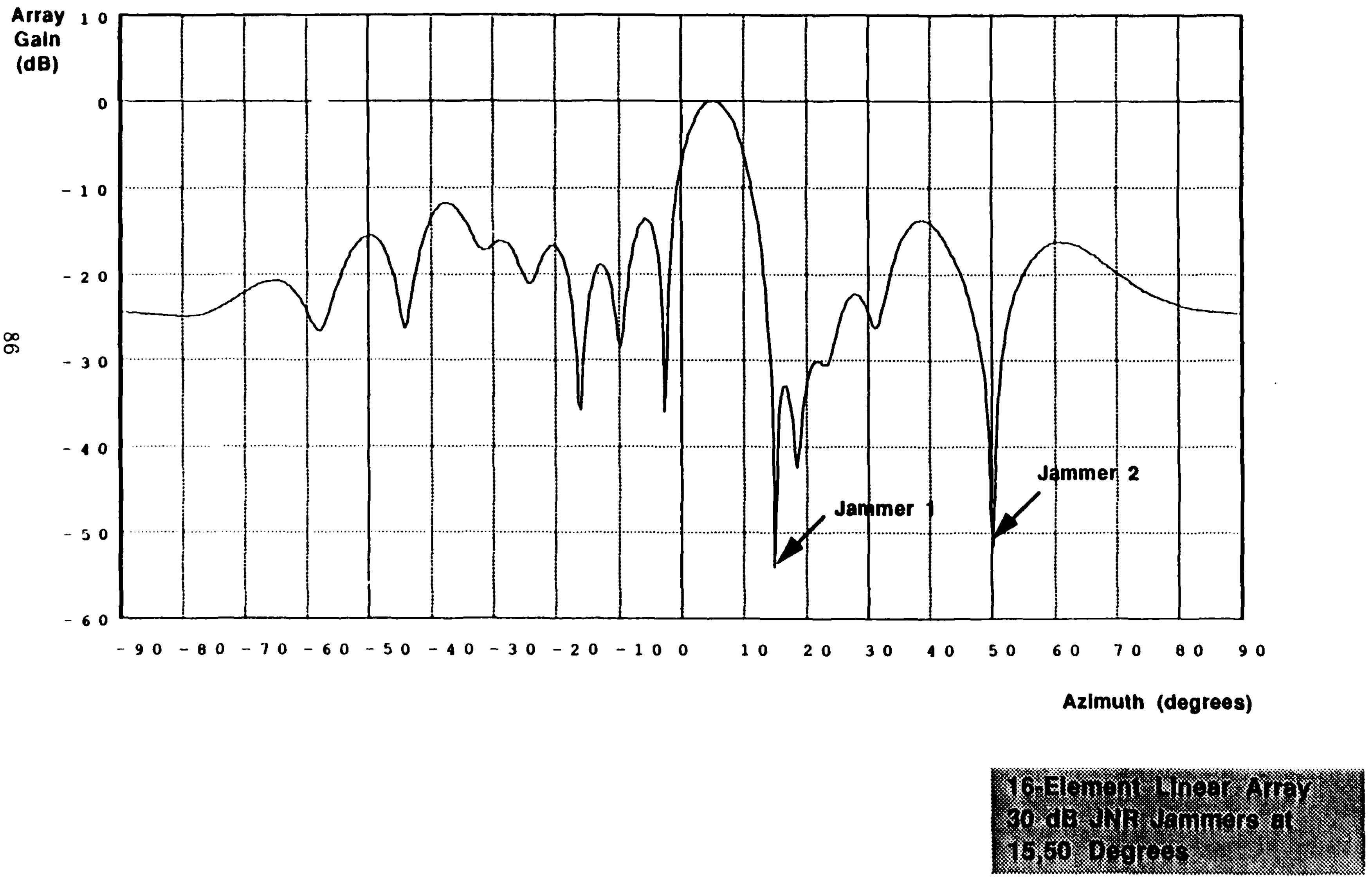
so that

$$\vec{w} = \mu R_{xx}^{-1} \vec{u}_0 = \mu [E_s(\Lambda_s^{-1} - \sigma^{-2} I) E_s^H + \sigma^{-2} I] \vec{u}_0 \tag{3.18}$$

Example 3.3

Figures 3.8-3.11 illustrate the foregoing points. Figures 3.8 and 3.9 depict the normalized antenna patterns for the weight vector of Example 3.1 for $N = 100$ snapshots for the case of no user signal, and a user-present signal having a 30 dB SNR. Clearly both patterns exhibit nulls near the jammers. However, the pattern of Figure 3.9 shows that when the

Figure 3.8: Antenna Pattern After 100 Snapshots Without User Present



user signal is introduced, a null also is placed near the user. Consequently very large peaks result when the pattern is scaled to produce unit gain to the user.

Figures 3.10 and 3.11 portray the situation in a different manner. The figures depict the zero patterns for the z-plane polynominal

$$w(z) = \sum_{k=1}^{16} w_k^* z^{-k} \tag{3.19}$$

where w_k^* denotes the k^{th} sensor weights and z is related to physical arrival angle by

$$z = e^{j\pi \sin\theta} \tag{3.20}$$

Figure 3.10 shows that with no user signal present zeros occur near the z values corresponding to jammer directions, but no zero occurs near the z value for the user direction. Figure 3.11 shows that when the user signal is introduced a zero is moved to the immediate vicinity of the z value for the user.

End of Example 3.3

3.2.6 Removal of Null on the User

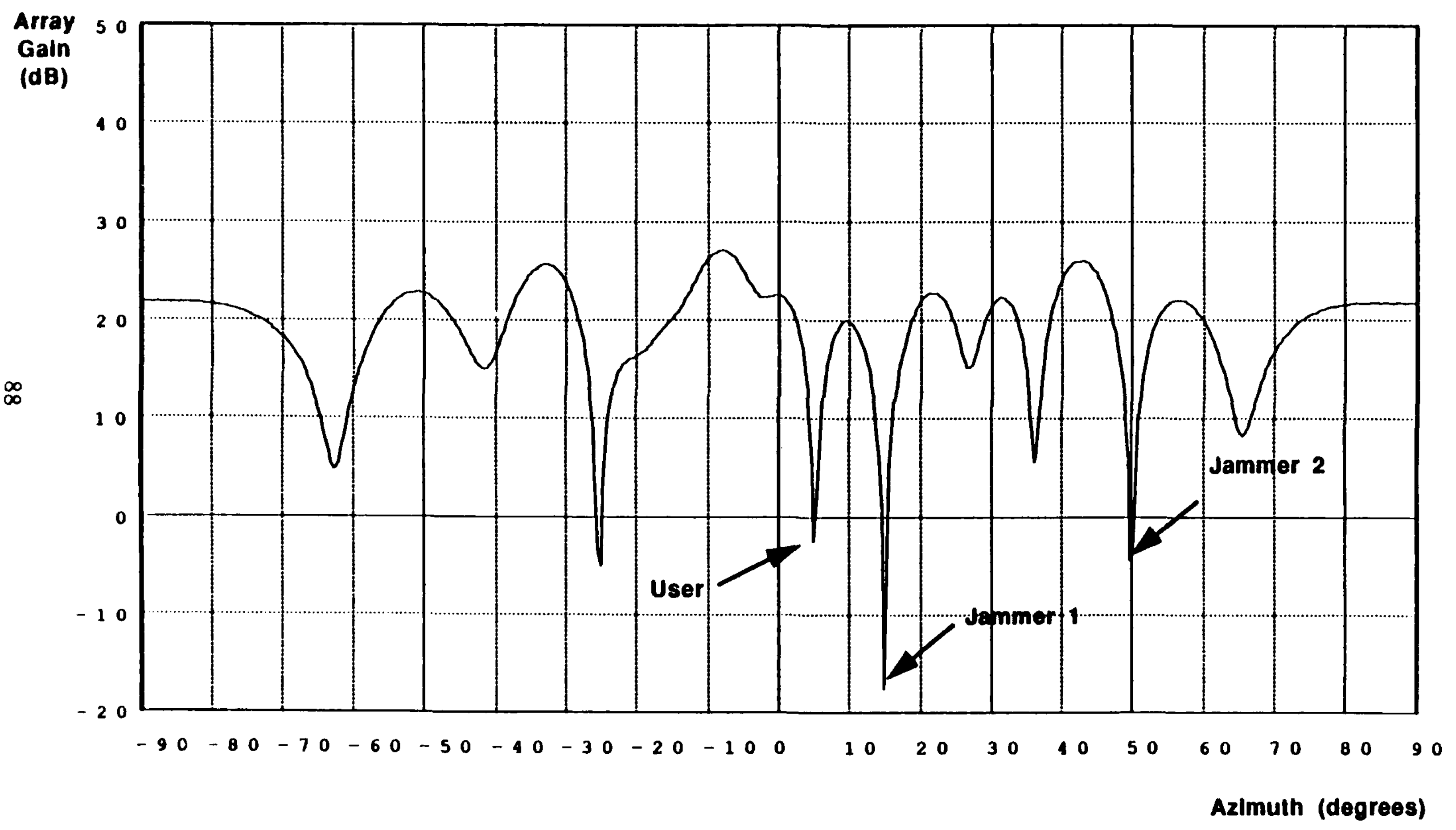
The foregoing example suggests a possible modification to accelerate the convergence of SMI when the user signal is present. The modification involves the z-plane polynomial (3.19). As noted, the poi,nomial (3.19) has zeros not only near the jammer, but also near the user. Since the null near the user causes the problem, the modification simply is to find and "divide out" the troublesome zero, leaving a polynomial (weight vector) that has no zero near the user but which does have zeros near the jammers. The following example illustrates this procedure.

Example 3.4

Figures 3.12 and 3.13 illustrate the results of using the modified SMI weight vector for the scenario of Examples 3.1 and 3.2.

Figure 3.12 depicts the array output SINR for the weight vector which results after removal of the zero near the user. To facilitate comparison with SMI, the curves of Figure 3.4 are reproduced in Figure 3.12. Clearly, *the resultant weight vector produces 30-40 dB more suppression than the unmodified SMI weight vector for $N \leq 100$, and produces suppression comparable to that of the SMI weight vector for the no-user case.*

Figure 3.9: Antenna Pattern After 100 Snapshots With User Present



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15,50 Degrees

Figure 3.10: Locations of Zeros in Complex Z-Plane After 100 Snapshots Without User Present

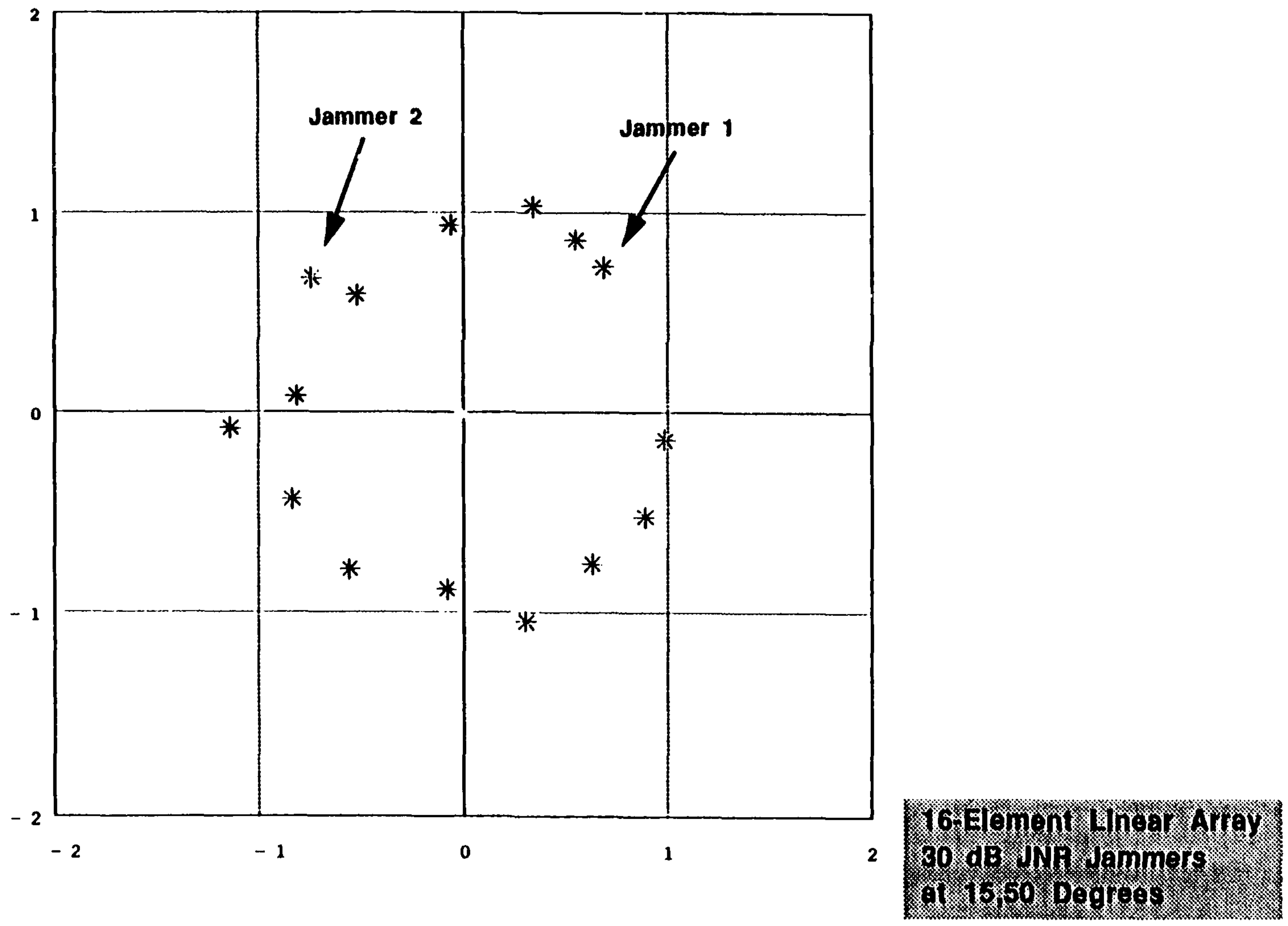
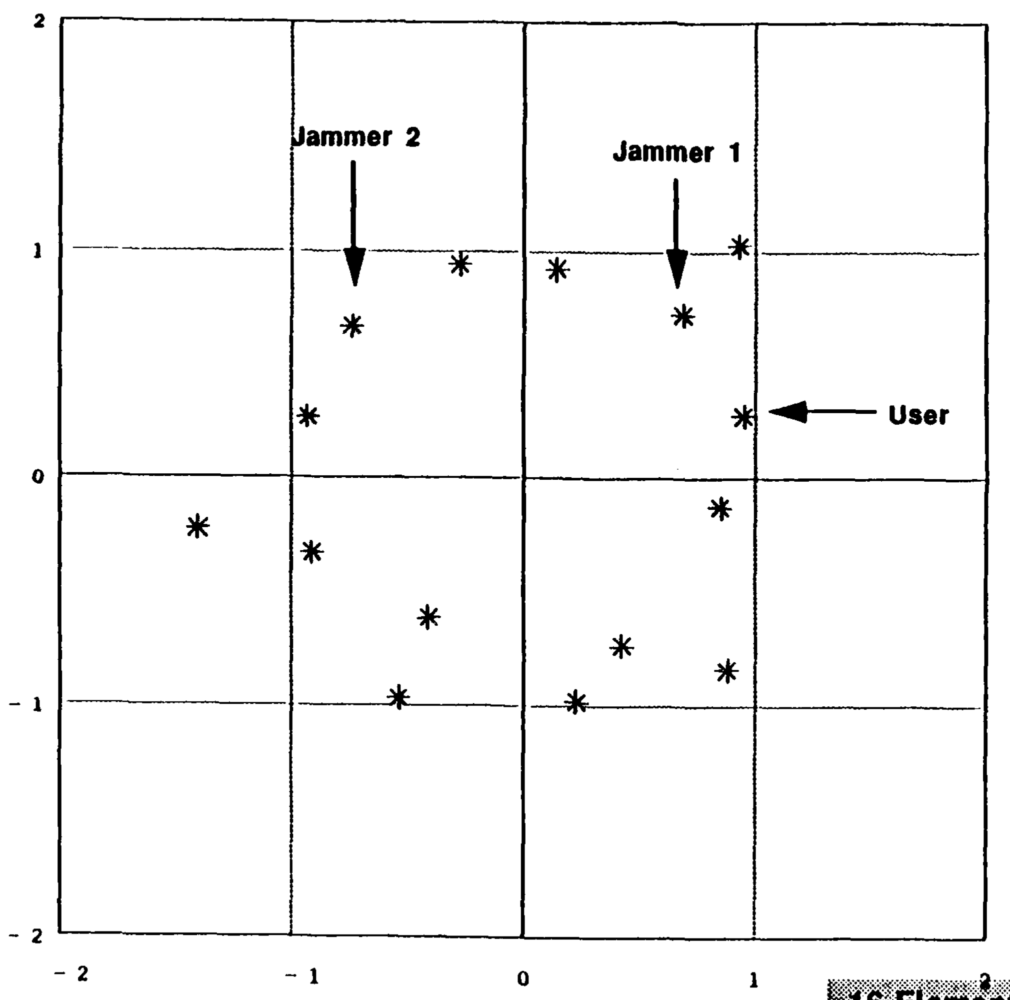


Figure 3.11: Locations of Zeros in Complex Z-Plane After 100 Snapshots With 30 dB SNR User Present



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15,50 Degrees

Figure 3.12: Improvement in Transient Response Due to Removal of Zero Near User

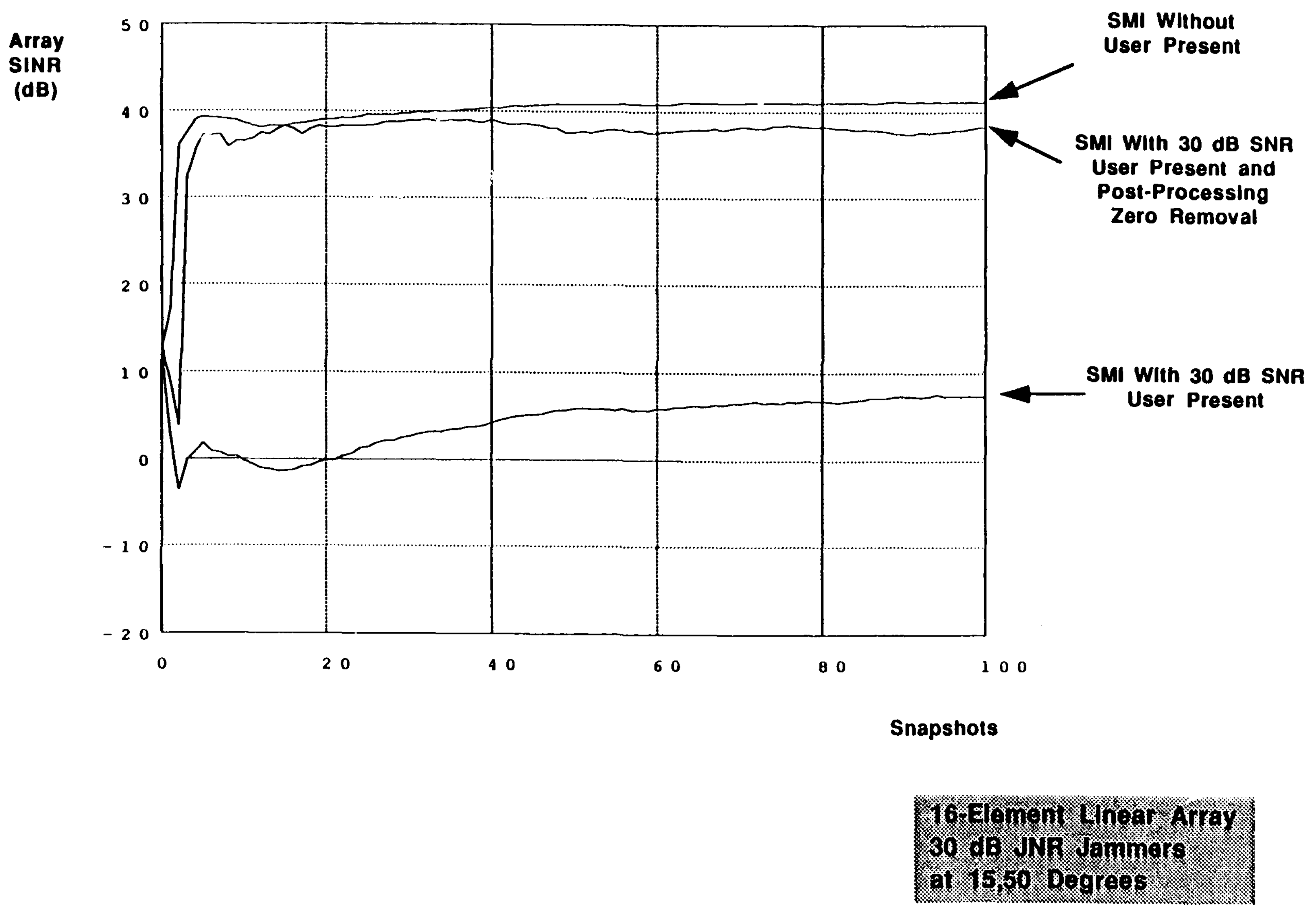


Figure 3.13 depicts the antenna pattern for the modified weight vector after 100 snapshots. While the pattern differs from the asymptotic pattern of Figure 3.3, the pattern nonetheless represents a major improvement over the pattern of Figure 3.9.

End of Example 3.4

Although the foregoing modification can improve transient performance in the case of a uniform linear array, it becomes cumbersome in the case of broadband arrays, and breaks down completely for the case of two-dimensional and three-dimensional arrays. The difficulty is due to the fact that the zeros of multidimensional arrays are contours, as opposed to single points, and zero division thus requires removal of the entire contour. In most cases, this is not possible since multidimensional polynomials are not factorizable into the product of smaller polynomials. Additionally, the adaptive processor places a null *nearby the user*, as opposed to directly on the user. Thus, the multidimensional procedure is required to both determine the zero contour which traces out a path nearby the user, and then to remove the corresponding zero-crossing contour.

Nonetheless, in Section 3.3 we describe a class of algorithms which exploit the essence of the foregoing modification. The thrust of the algorithms is to *constrain* the adaptive processor so as to:

1. Ensure factorability of the weight vector.
2. Place nulls *exactly* on the users.

The (known) factor that nulls the users then can be "divided out" leaving a final weight vector that passes users and nulls jammers.

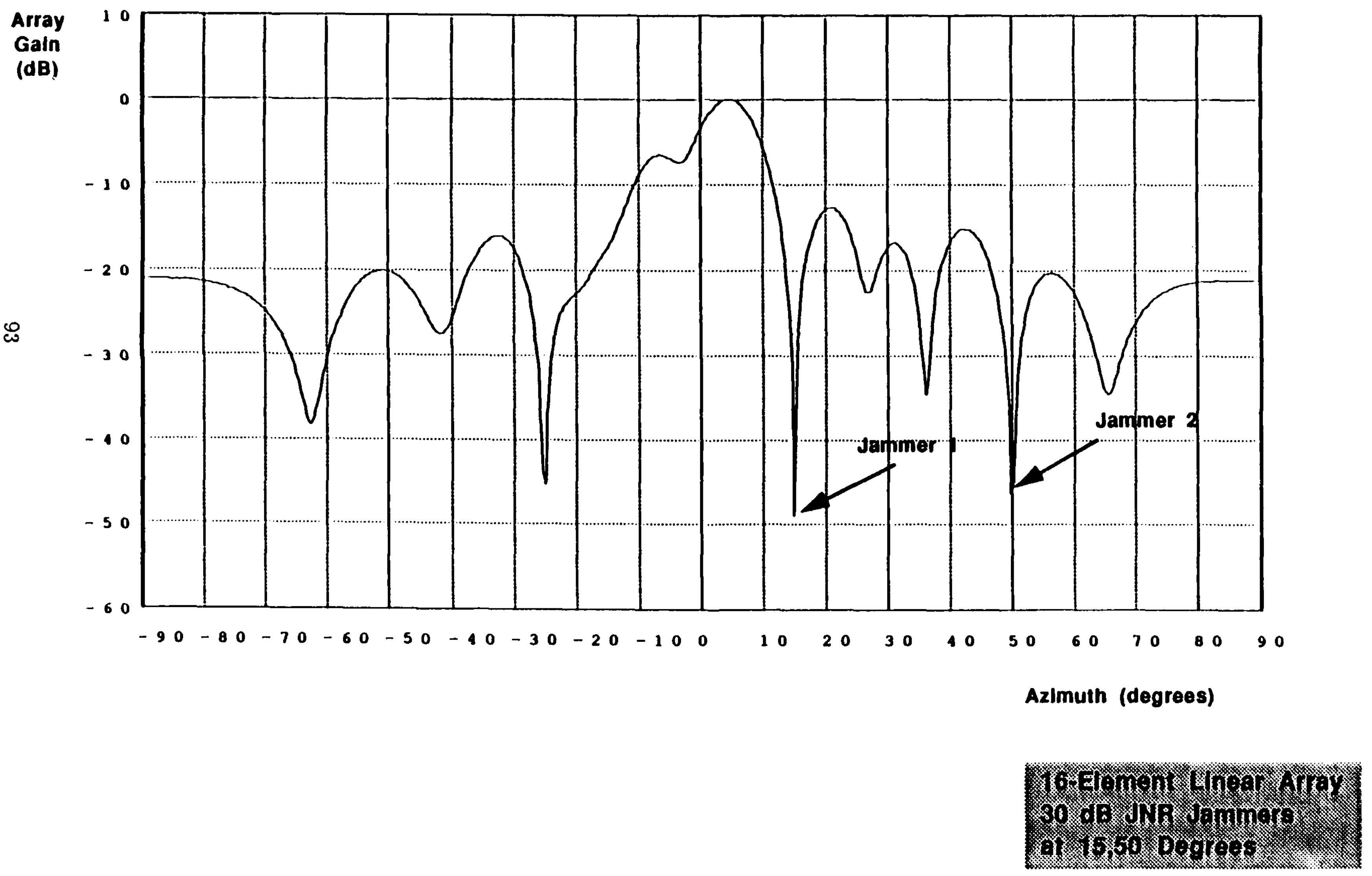
3.3 New Class of Nulling Algorithms

To counter the adverse effects of user signals in adaptive arrays, we have developed a new class of nulling algorithms. The algorithms use signal processing techniques to spatially filter the user signal prior to weight determination, and to provide a final weight vector which satisfies a system of Q linear equations of the form

$$C^H\vec{w} = \vec{f} \tag{3.21}$$

where C is a $K \times Q$ constraint matrix, and $\vec{f}$ is a $Q \times 1$ constraint vector. The system (3.21) is a generalization of the single gain constraint. As described in Section 2.4, the constraints

Figure 3.13: Antenna Pattern After 100 Snapshots With User Signal Present and Post-Processing Zero Removal



can be selected to achieve a variety of design objectives including preservation of gain to multiple users, preservation of gain across multiple frequencies, control of various pattern derivatives, etc. As we will show, the new class of algorithms can greatly reduce the adverse effects of user signal presence as described in Section 3.2.

3.3.1 Basic Idea

The new class of algorithms exploits the convolutional array structure and consists of the following two steps: 1) All-Emitter Nulling, 2) User Restoration. The algorithms utilize a pair of variable weight-and-sum processors as shown in Figure 3.14. This architecture generalizes a structure due to Duvall[4].

The all-emitter nulling step is performed by cascading weight-and-sum processors having weight sets $\vec{w}_{\overline{U}}$ and $\vec{w}_J$ as illustrated in the top portion of Figure 3.14. This step involves constructing a full-aperture weight set $\vec{w}_N$ which can be expressed as the *convolution* of the smaller weight sets $\vec{w}_{\overline{U}}$ and $\vec{w}_J$. That is

$$\vec{w}_N = \vec{w}_{\overline{U}} \star \vec{w}_J \tag{3.22}$$

where $\star$ denotes convolution across one, two, or three dimensions.

Ideally, $\vec{w}_{\overline{U}}$ is a (small) fixed weight set which is designed to:

$\overline{U}1$: Null the user(s).

$\overline{U}2$: Pass the jammer signals with high gain to provide high JNR.

The weight set $\vec{w}_{\overline{U}}$ is designed based on a priori knowledge of the user direction(s), and coarse information concerning the number and directions of the jammers. The directional information can be obtained through coarse direction-of-arrival analysis (in some cases through use of the DFT or FFT). Once $\vec{w}_{\overline{U}}$ has been selected, the weight set $\vec{w}_J$ is determined so that the composite weight vector given in (3.22) minimizes the time-averaged power subject to constraints upon the *final* weight set (yet to be described). The power minimization operation determines the weight set $\vec{w}_J$ such that the final weight set places nulls on the jammers. The constraints prevent the weight set $\vec{w}_J$ from placing nulls on the users.

The user restoration step is performed by cascading weight-and-sum processors having weight sets $\vec{w}_U$ and $\vec{w}_J$ as illustrated in the bottom portion of Figure 3.14. This step involves

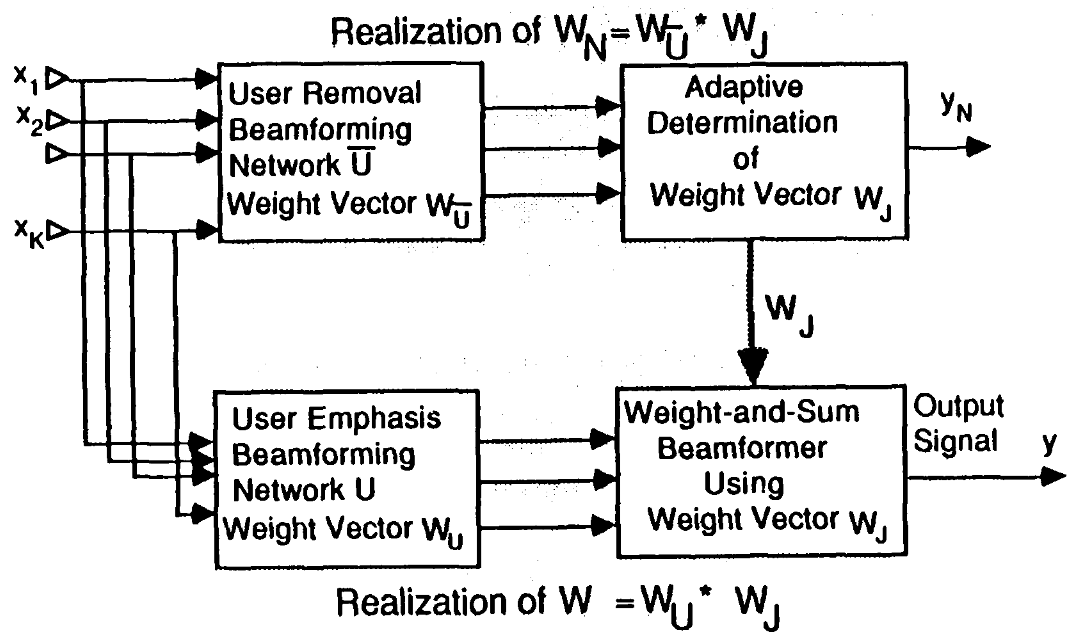

Realization of $W_N = W_{\overline{U}} * W_J$
x_1
x_2
x_K
User Removal Beamforming Network $\overline{U}$ Weight Vector $W_{\overline{U}}$
Adaptive Determination of Weight Vector W_J
y_N
W_J
User Emphasis Beamforming Network U Weight Vector W_U
Weight-and-Sum Beamformer Using Weight Vector W_J
Output Signal
y
Realization of $W = W_U * W_J$

Figure 3.14: Architecture of New Class of Nulling Algorithms

constructing a full aperture weight set $\vec{w}$ which can be expressed as the *convolution* of the smaller weight sets $\vec{w}_U$ and $\vec{w}_J$, that is

$$\vec{w} = \vec{w}_U \star \vec{w}_J \tag{3.23}$$

The weight set $\vec{w}_U$ is complementary to the set $\vec{w}_{\overline{U}}$ in that it is designed to:

$U1$: Pass the users.

$U2$: Provide some suppression of jammers without requiring explicit jammer directions (e.g. via sidelobe control).

The far-field antenna pattern of the weight set in (3.23) is the product of the patterns corresponding to weight sets $\vec{w}_U$ and $\vec{w}_J$. Because of the manner in which the weight sets $\vec{w}_U$ and $\vec{w}_J$ are constructed, the composite weight set in (3.23) passes the users and nulls the jammers.

3.3.2 Algorithm Equations

To specify the class of algorithms more precisely, we next present general expressions for implementing the algorithm, and illustrate the equations for an Example Scenario characterized by the following assumptions:

- The array is a linear phased-array consisting of K equally-spaced omni-directional elements.
- All signals are narrowband; consequently a weight-and-sum beamformer is employed.
- A single user impinges on the array from the broadside direction so that

$$\vec{u}_0 = [1, 1 \cdots 1]^T \tag{3.24}$$

- The processor weight vector $\vec{w}$ provides unit gain in the direction of the broadside user, that is

$$\vec{u}_0^H \vec{w} = 1 \tag{3.25}$$

- The filter $\vec{w}_{\overline{U}}$ is a dipole which nulls the user, that is

$$\vec{w}_{\overline{U}} = [1, -1]^T \tag{3.26}$$

- The filter $\vec{w}_U$ adds signals from adjacent pairs of sensors, that is

$$\vec{w}_U = [1,1]^T \tag{3.27}$$

Note that the resultant processing structure for the Example Scenario is closely related to the Duvall Beamformer.

For purposes of developing the equations, assume that $\vec{w}_N$ and $\vec{w}$ are $K \times 1$ vectors, and that $\vec{w}_J$ is a $\overline{K} \times 1$ vector where $\overline{K} < K$. Thus, for the example scenario $\overline{K} = K - 1$.

The two-level filtering operation, depicted in Figure 3.14, amounts to expressing the weight vector $\vec{w}_N$ in terms of the weight vector $\vec{w}_J$ as follows:

$$\vec{w}_N = \overline{U}\vec{w}_J \tag{3.28}$$

where $\overline{U}$ is a $\overline{K} \times K$ matrix, the elements of which are determined from the weight set $\vec{w}_{\overline{U}}$, and the requirement that equation (3.1) is satisfied. For the Example Scenario, the $K \times (K-1)$ matrix $\overline{U}$ is written as

$$\overline{U} = \begin{pmatrix} 1 & 0 & 0 & \cdot & \cdot & 0 \\ -1 & 1 & 0 & \cdot & \cdot & 0 \\ 0 & -1 & 1 & \cdot & \cdot & 0 \\ \cdot & & & & & \\ \cdot & & & & & \\ 0 & \cdot & \cdot & 0 & -1 & 1 \\ 0 & \cdot & \cdot & \cdot & 0 & -1 \end{pmatrix} \tag{3.29}$$

The representation in (3.28) can be viewed as passing the $K \times 1$ array snapshot $\vec{x}(n)$ through a beamformer having a beamformer matrix $\overline{U}$ to produce a $\overline{K} \times 1$ beamspace snapshot, i.e.

$$\vec{\overline{x}}(n) = \overline{U}^H \vec{x}(n) \tag{3.30}$$

since

$$\vec{w}_N^H \vec{x}(n) = \vec{w}_J^H \overline{U}^H \vec{x}(n) = \vec{w}_J^H \vec{\overline{x}}(n) \tag{3.31}$$

To facilitate power minimization, we first represent the weight $\vec{w}_J$ vector as follows:

$$\vec{w}_J = M\vec{w}_V \tag{3.32}$$

where M is a $\overline{K} \times K'$ beamformer matrix, $\vec{w}_V$ is a $K' \times 1$ vector, and

$$K' \leq \overline{K} \tag{3.33}$$

The elements of $\vec{w}_V$ represent the parameters that are allowed to vary in order to minimize power. In the simplest case $M = I$ so that $\vec{w}_J = \vec{w}_V$, and all elements of the weight vector w_J are treated as variable. We show in Section 3.4 that it is advantageous to select M in a different manner - one which exploits a-priori information. Specifically, it is beneficial to select M to be a $\overline{K} \times (J+1)$ matrix, the columns of which are weight vectors that steer beams in the approximate directions of the $J+1$ emitters. However, for purposes of the Example Scenario we select

$$M = I \tag{3.34}$$

Use of the representations (3.28) and (3.32) is equivalent to expressing $\vec{w}_N$ as

$$\vec{w}_N = \overline{U} M \vec{w}_V \tag{3.35}$$

The time-averaged power output power for the array of Figure 3.14 is

$$p = E\{|y|^2\} = \vec{w}_N^H \hat{R}_{xx} \vec{w}_N = \vec{w}_V^H \hat{R}'_{xx} \vec{w}_V \tag{3.36}$$

where

$$\hat{R}'_{xx} = M^H \overline{U}^H \hat{R}_{xx} \overline{U} M \tag{3.37}$$

and $\hat{R}_{xx}$ denotes the time-averaged covariance matrix.

The weight vector $\vec{w}$ used in the composite weight-and-sum processor in Figure 3.14 is related to $\vec{w}_J$ by an expression of the form

$$\vec{w} = U \vec{w}_J \tag{3.38}$$

where U is a $K \times \overline{K}$ matrix, the elements of which are determined from the weight set $\vec{w}_U$ and the requirement that $\vec{w}$, $\vec{w}_U$ and $\vec{w}_J$ satisfy (3.8). For the example scenario, U is a $K \times (K-1)$ matrix given by

$$U = \begin{pmatrix} 1 & 0 & 0 & \cdot & \cdot & 0 \\ 1 & 1 & 0 & \cdot & \cdot & 0 \\ 0 & 1 & 1 & \cdot & \cdot & 0 \\ \cdot & & & & & \\ \cdot & & & & & \\ \cdot & \cdot & \cdot & 0 & 1 & 1 \\ 0 & \cdot & \cdot & \cdot & 0 & 1 \end{pmatrix} \tag{3.39}$$

Use of the representations (3.32) and (3.38) is equivalent to representing $\vec{w}$ as

$$\vec{w} = U M \vec{w}_V = B \vec{w}_V \tag{3.40}$$

where

$$B \equiv UM \tag{3.41}$$

If $\vec{w}$ is to satisfy constraints of the form

$$C^H \vec{w} = \vec{f} \tag{3.42}$$

then the corresponding constraint on $\vec{w}_V$ is

$$C'^H \vec{w}_V = \vec{f} \tag{3.43}$$

where

$$C' = M^H U^H C \tag{3.44}$$

For the Example Scenario, equation (3.42) takes the form

$$[1, 1 \cdot \cdot 1]\vec{w} = 1 \tag{3.45}$$

so that C is a single column of K elements

$$C = [1, 1 \cdot \cdot 1]^T \tag{3.46}$$

and

$$\vec{f} = 1 \tag{3.47}$$

Use of (3.34), (3.39) and (3.46) in (3.44) yields C' as a column of $K - 1$ elements

$$C' = 2[1, 1 \cdot \cdot 1]^T \tag{3.48}$$

The weight vector $\vec{w}_V$ which minimizes the power expression (3.36) subject to the constraint (3.43) is given by

$$\vec{w}_V = \hat{R}'^{-1}_{xx} C' [C'^H \hat{R}'^{-1}_{xx} C']^{-1} \vec{f} \tag{3.49}$$

where

$$R'_{xx} = M^H \overline{U}^H \hat{R}_{xx} \overline{U} M \tag{3.50}$$

and

$$C' = M^H U^H C \tag{3.51}$$

Thus, the composite weight vector $\vec{w}$ for the processor of Figure 3b is

$$\vec{w} = B\vec{w}_V = UM\hat{R}'^{-1}_{xx} C' [C'^H \hat{R}'^{-1}_{xx} C']^{-1} \vec{f} \tag{3.52}$$

For the Example Scenario, the weight vector is

$$\vec{w} = \mu U[\overline{U}^H \hat{R}_{xx}\overline{U}]^{-1}U^H\vec{u}_0 \tag{3.53}$$

$$\mu = 1/\vec{u}_0^H U[\overline{U}^H \hat{R}_{xx}\overline{U}]^{-1}U^H\vec{u}_0 \tag{3.54}$$

For the single constraint case equation (3.52) can also be written as

$$\vec{w} = \mu UM[M^H\overline{U}^H \hat{R}_{xx}\overline{U}M]^{-1}M^HU^H\vec{u}_0 \tag{3.55}$$

where

$$\mu = 1/\vec{u}_0^H UM[M^H\overline{U}^H \hat{R}_{xx}\overline{U}M]^{-1}M^HU^H\vec{u}_0 \tag{3.56}$$

3.3.3 Relationship to the Duvall Beamformer

The processing structure of Figure 3.14 can be regarded as a generalization of a structure developed by Duvall to reduce distortion in broadband nulling systems[4]. In the case of the of the Duvall structure:

1. The sensors are placed in a uniform linear array.
2. The pre-processing filter is a dipole; that is, for a broadside user $\vec{w}_{\overline{U}} = (1,-1)^T$.
3. The adaptive processor consists of a Frost beamformer with a maximum number of degrees of freedom.
4. The weight set $\vec{w}_J$ is used *directly* in the slave processor; that is, the weight set $\vec{w}_U = (1,0)$ and therefore is omni-directional.

Our processing structure generalizes the Duvall Beamformer in the following ways:

1. The sensors can be placed in one, two or three-dimensional convolutional arrays (vs. uniform linear arrays).
2. The pre-processing filter with weight set $\vec{w}_{\overline{U}}$ may possess more than one degree of freedom (which facilitates removal of multiple users, and removal of user signals from multidimensional arrays).
3. The adaptive processor may utilize thinned beamspace and other techniques to improve performance and reduce computation (vs. the full Frost beamformer).
4. The prefilter with weight set $\vec{w}_U$ is added to the slave processor to provide additional jammer suppression.

3.3.4 Broadband Algorithms

Although we have described the new class of algorithms in the context of narrowband arrays, the algorithms also apply to broadband arrays provided that the convolution operations in equations (3.22) and (3.23) are properly interpreted. In the narrowband case, the operation consists of performing a a one-dimensional (or linear) convolution of two linear weight sets to obtain a third linear weight set. In the case of a broadband array, the weights are defined as two-dimensional as illustrated in Figure 3.15a. One dimension (k) corresponds to the sensor index, and the second dimension (l) corresponds to the tap on the delay-line to which the weight $w_{l,k}$ is applied.

Thus, for broadband arrays, the operations in (3.22) and (3.23) are *two-dimensional convolutions.* That is,

$$(\vec{w}_N)_{l,k} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (\vec{w}_{\overline{U}})_{i,j}(\vec{w}_J)_{l-i,k-j} \tag{3.57}$$

and

$$(\vec{w})_{l,k} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (\vec{w}_U)_{i,j}(\vec{w}_J)_{l-i,k-j} \tag{3.58}$$

Figure 3.15b illustrates the points of definition (or support) for a pair of the possible conventional factors for the weight set of Figure 3.15a.

In broadband problems, the fixed weight sets $\vec{w}_{\overline{U}}$ and $\vec{w}_U$ are defined on (small) rectangular point sets to achieve objectives $\overline{U1}$, $\overline{U2}$, $U1$, and $U2$ discussed in Section 3.3.1. The non-fixed weight set $\vec{w}_J$ is defined on a complementary rectangular point set. The weights $\vec{w}_J$ are varied via the variable weight set $\vec{w}_V$ so as to minimize the array output power, while ensuring that the final weight set $\vec{w}$ satisfies the linear constraints (3.42). Once again the final weight vector is given by (3.52) provided that all the matrices are interpreted in the context of stacked snapshots and stacked weight vectors (See Section 2.3.1).

3.4 One-Dimensional Examples of the New Algorithms

A variety of specific algorithms can be generated by selecting the weight set $\vec{w}_{\overline{U}}$ to be a two-point filter that nulls the user (i.e. a dipole), and by appropriately specializing the matrix M in equations (3.35)-(3.52). We have examined a variety of choices for one-dimensional arrays. Example possibilities are described as follows.

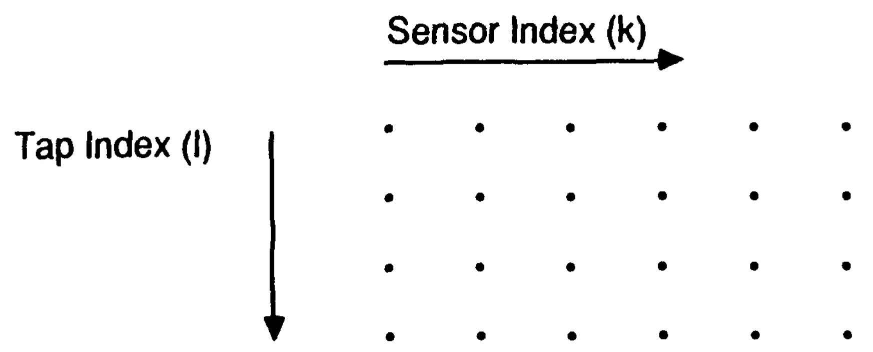


Figure 3.15a: Example grid points for two-dimensional weight set $\overline{W}_N$ for a linear broadband array.

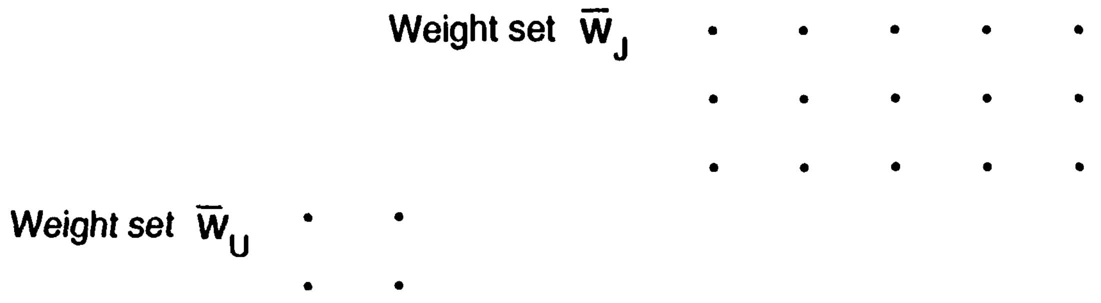


Figure 3.15b: Example grid points for two factor weight sets $\overline{W}_J$ and $\overline{W}_U$ for a linear broadband array

Algorithm 1

Select $M = I$ so that $\vec{w}_J = \vec{w}_V$. This selection provides $K-1$ degrees of freedom, and also requires the inversion of the $(K-1) \times (K-1)$ matrix R'_{xx}in (3.50).

Algorithm 2

Select M so that the weight set $\vec{w}_J$ is the convolution of a designed fixed weight set $\vec{w}_a$ and a variable weight set $\vec{w}_V$; that is

$$\vec{w}_J = \vec{w}_a \star \vec{w}_V \tag{3.59}$$

The weight set $\vec{w}_a$ is designed to "pass" all emitters based upon approximate directional information, and to otherwise be of minimum norm. Thus, if the weight set $\vec{w}_a$ consists of the components $a_1, a_2, \cdot\cdot a_q$ then the matrix M has dimensions of $(K-1) \times (K-q)$ and is of the form

$$M = \begin{pmatrix} a_1 & 0 & \cdot & \cdot & \cdot & 0 \\ a_2 & a_1 & 0 & \cdot & & 0 \\ \cdot & \cdot & \cdot & & & \cdot \\ a_q & \cdot & & & & a_1 \\ 0 & a_q & \cdot & & & a_2 \\ \cdot & 0 & & & & \cdot \\ \cdot & \cdot & & & a_q & \cdot \\ 0 & 0 & \cdot & & & a_q \end{pmatrix} \tag{3.60}$$

This selection of M makes available $(K-q-1)$ degrees of freedom and requires inversion of a $(K-q) \times (K-q)$ matrix R'_{xx}.

Algorithm 3

Select M to be a $(K-1) \times (J+1)$ matrix of the form

$$M = [\vec{v}_0, \vec{v}_1, \cdot\cdot \vec{v}_J] \tag{3.61}$$

where $\vec{v}_0, \vec{v}_1, \cdot\cdot \vec{v}_J$ are weight vectors which steer beams in the approximate directions of the $J+1$ emitters. This selection makes available the minimum number $J+1$ degrees of freedom sufficient to null the J jammers, and requires inversion of the $(J+1) \times (J+1)$ matrix R'_{xx}.

Collectively, the preceding algorithms span an interesting set of alternatives. Algorithm 1 makes available the maximum number of degrees of freedom, and requires the greatest amount of computat.on. Algorithm 3 provides smallest number of degrees of freedom and requires the least amount of computation. Algorithm 2 is intermediate between Algorithms 1 and 3.

Performance-wise, *all three algorithms have shown dramatic improvements in the transient response when the user signal is present as compared to conventional SMI*, for the examples we have examined. There is little difference in performance of the algorithms for sidelobe jamming scenarios. For mainbeam jamming scenarios, the algorithms exhibit different performance properties. Algorithm 3 seems to have the best performance if the beams can be pointed on, or very near the emitters. Algorithm 2 provides the best performance for a filter length which is approximately equal to half the array aperture. Algorithm 1 seems to perform better than algorithms 2 and 3. The selection between algorithms 1, 2 and 3 represents a tradeoff between performance and computation. Further quantification and algorithm selection is an important area for further work.

The following examples demonstrate the performance improvements of the class of algorithms.

Example 3.5

Consider again the sidelobe jamming scenario of Example 3.1. Algorithms 1 and 3 were used to generate weight vectors for the case in which the 30 dB SNR user was present. In the case of Algorithm 3, the weight vectors $\vec{v}_0, \vec{v}_1$ and $\vec{v}_2$ were selected to steer beams one degree away from the actual emitter directions.

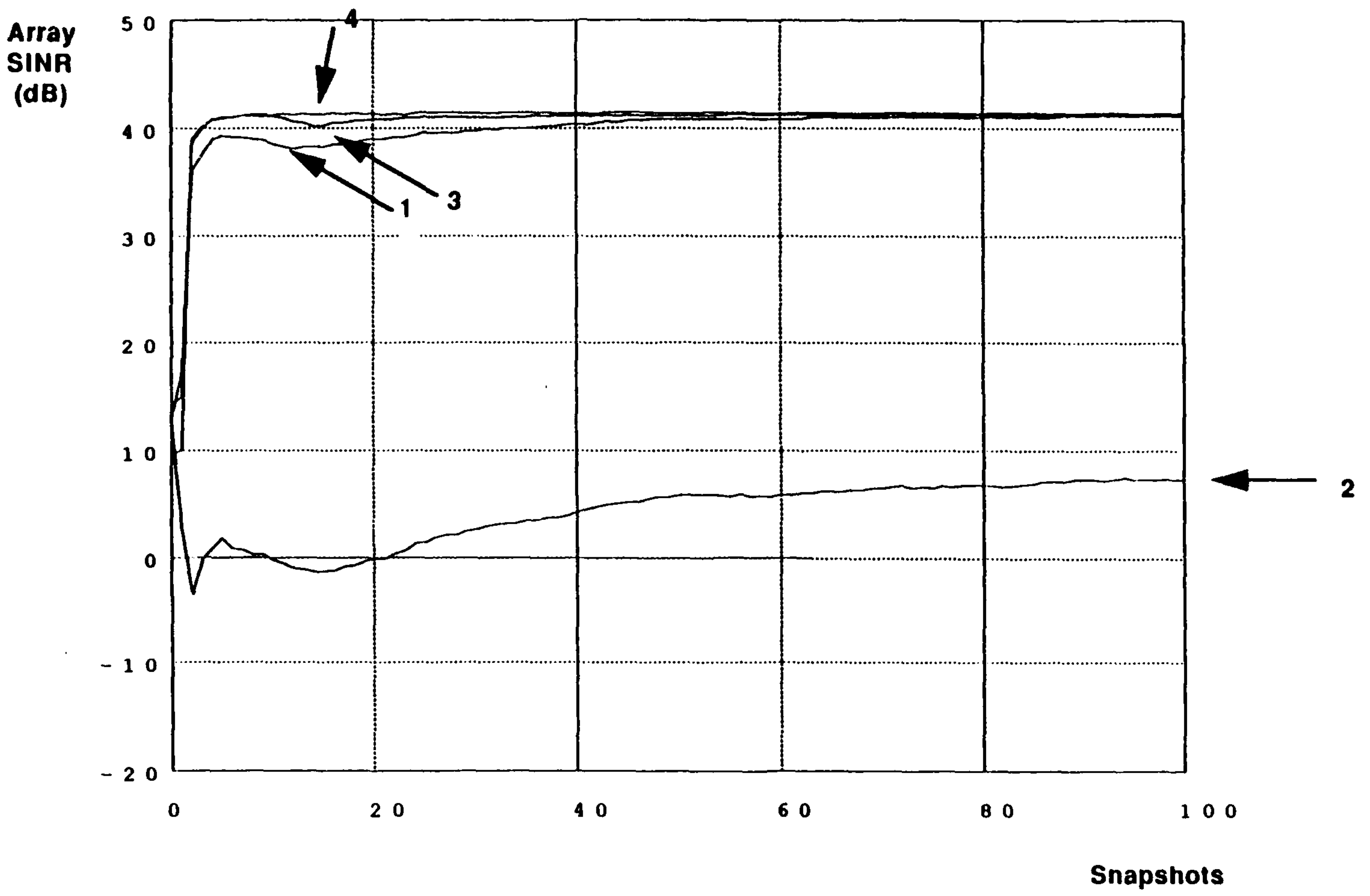
The SINR versus N results for each weight vector are plotted as Curves 4 and 5 in Figure 3.16. For reference, the curves of Figure 3.7 are also re-plotted as Curves 1-3. Comparison of the curves shows that *the transient response of both algorithms with the user present exceeds that of SMI with the user present by 30-40 dB for* $N \leq 100$.

Figure 3.17 depicts the antenna pattern for the weight vector (3.53) for $N = 100$. Comparison of Figure 3.17 with Figures 3.3 and 3.13 shows that the antenna pattern for Algorithm 3 is much closer to the steady-state pattern of Figure 3.3 than it is to the coarser pattern of Figure 3.13.

End of Example 3.5

Example 3.6

Figure 3.16: Improvement in Transient Response Due to Convolutional Filtering Algorithms 1, 3



Curve 1: SMI Without User Present

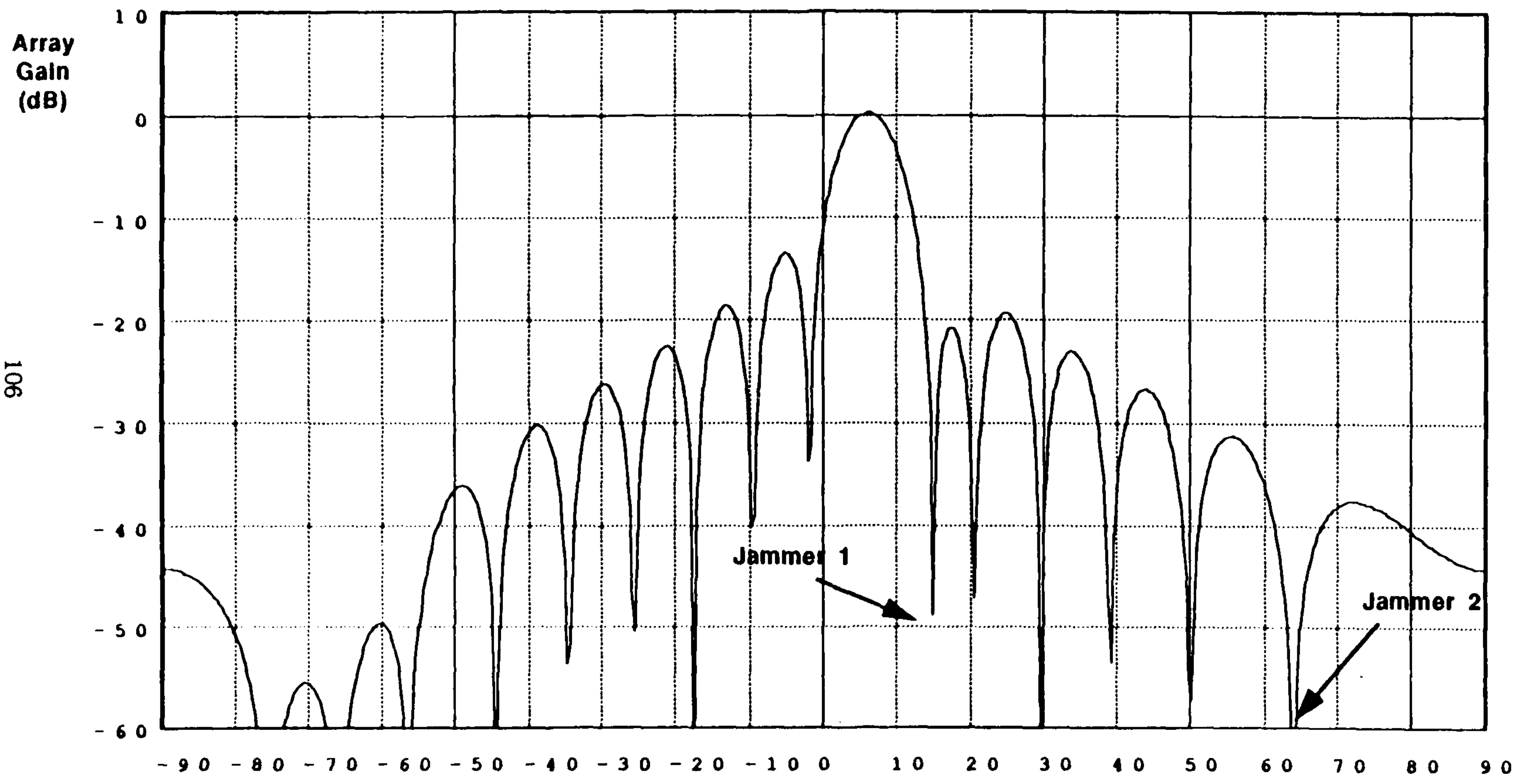
Curve 2: SMI With 30 dB SNR User Present

Curve 3: Algorithm 1 (New) With 30 dB SNR User Present

Curve 4: Algorithm 3 (New) With 30 dB SNR User Present

16-Element Linear Array
30 dB JNR Jammers
at 15,50 Degrees

Figure 3.17: Antenna Pattern After 100 Snapshots With User Signal Present and Use of Algorithm 3



16-Element Linear Array
30 dB SNR Users at 5 Degrees
30 dB JNR Jammer at 15,50 Degrees

Algorithm 2 was implemented for the sidelobe jamming scenario of Example 3.1 for the cases of $q = 6, 12$. In both cases, the filter weights $a_1, a_2, \cdots a_q$ were selec'ed to provide unit gain in the directions of the emitters and to otherwise exhibit minimum norm.

Figure 3.18 depicts the performance of the algorithm. The SINR versus N results are plotted as Curves 3 and 4. Once again the transient response with the 30 dB user present is comparable to that of conventional SMI with no user present, and 35-40 dB better than SMI with the user present for $N \leq 100$.

End of Example

Example 3.7

The scenario of Example 3.1 was modified by moving the jammer located on the quiescent sidelobe peak at 15° to a mainbeam location of 7°. The resulting scenario thus consists of a 30 dB SNR user at 5°, a 30 dB JNR (mainbeam) jammer at 7°, and a sidelobe jammer at 50°. The mainbeam jammer is now located less than a third of a beamwidth away from the user.

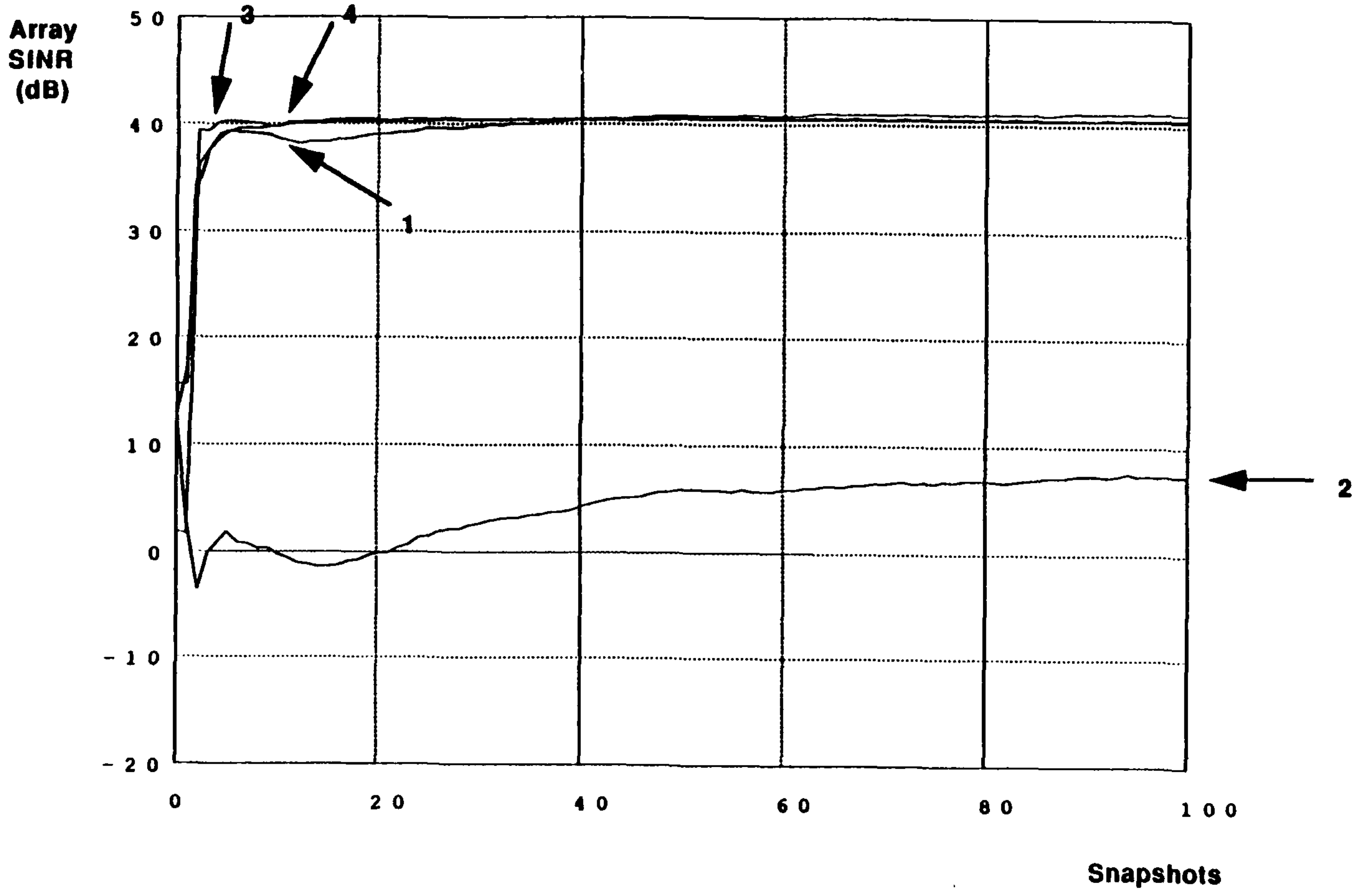
The transient response of Algorithms 1,2 and 3 was computed and compared with that of conventional SMI with the user absent/present. The results are shown in Figure 3.19. Curves 1 and 2 depict SINR versus N for the SMI algorithm, without and with the user present. Curves 3 and 4 depict array SINR versus N for algorithms 1 and 3. Curves 5,6 depict transient response for Algorithms 2 with $q = 6, 12$ respectively. The curves show that algorithms 1,2 and 3 exceed the performance of SMI when the user is present by at least 35 dB. Algorithm 2, with $q = 12$ exhibits an SINR which is down by about 9 db from the other algorithms. Algorithms 1, 2 ($q = 6$) and 3 are nearly equal in performance.

End of Example 3.7

Example 3.8

Algorithms which remove the user signal from the data substantially improve the transient response behavior but compromise performance for close-in mainbeam jamming scenarios. The compromise in performance is due to the fact that the jamming data is removed along with the user signal. Thus, the adaptive algorithm no longer senses the effects of the mainbeam jammer, and when the user signal is restored the jamming signal is also restored. The result is that the algorithms speed up the transient response but limit the achievable SINR to a value below that yielded by SMI without the user present.

Figure 3.18: Improvement in Transient Response Due to Convolutional Filtering Algorithm 2



Curve 1: SMI Without User Present

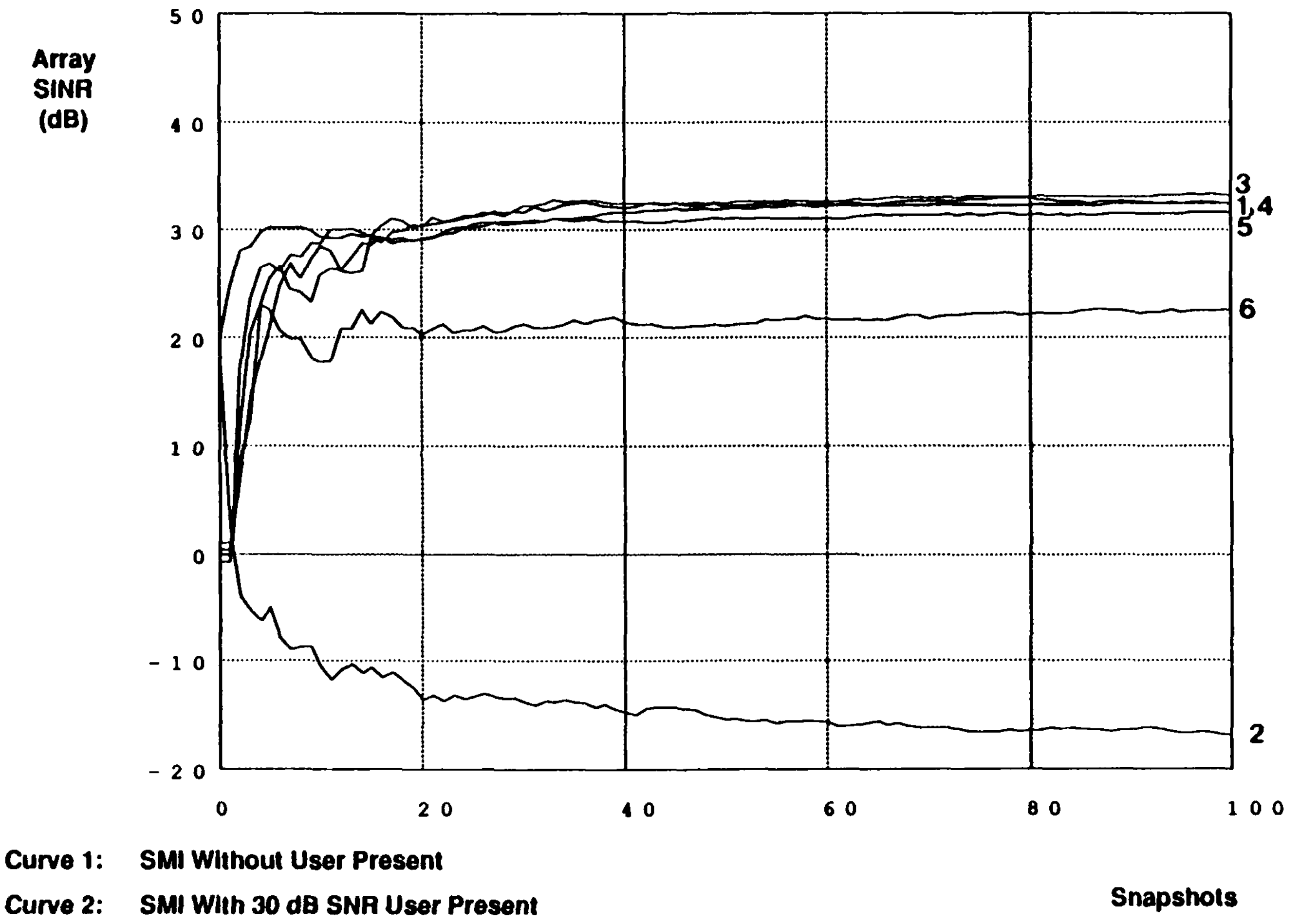
Curve 2: SMI With 30 dB SNR User Present

Curve 3: Algorithm 2 (6-Point Filter) With 30 dB SNR User Present

Curve 4: Algorithm 2 (12-Point Filter) With 30 dB SNR User Present

16-Element Linear Array
30 dB JNR Jammers
at 15,50 Degrees

Figure 3.19: Improvement in Transient Response
Due to Convolutional Filtering (Mainbeam, Sidelobe Jamming)



Curve 1: SMI Without User Present

Curve 2: SMI With 30 dB SNR User Present

Curve 3: Algorithm 1 With 30 dB SNR User Present

Curve 4: Algorithm 3 With 30 dB SNR User Present

Curve 5: Algorithm 2 (6-Point Filter) with 30 dB User Present

Curve 6: Algorithm 2 (12-Point Filter) with 30 dB User Present

16-Element Linear Array
30 dB JNR Jammers
at 7,50 Degrees

To see this effect more clearly, we computed the steady-state performance of Algorithm 1 with 2-point user removal and user emphasis filters as a function of the mainbeam jammer location. Curve 1 in Figure 3.20 depicts the steady-state SINR for SMI without the user present in the data used for adaptation. Curve 2 depicts the SINR for Algorithm 1. Each 0.5° interval represents 0.066 beamwidths. The curves show that the steady-state SINR gradually departs from that achievable with SMI (without the user present) as the jammer moves toward the user. The performance of the Algorithm 1 then re-approaches that of SMI as the jammer-to-user distance decreases below 5.5°.

It is felt that an improvement in this performance could be achieved by suitable modification of the algorithms. Additional simulations have shown that the performance in the face of close-in mainbeam jammers can be increased by different selection of beams in Algorithm 3, and a different choice of the constraint steering vector. An important topic for future work is the refinement and modification of the algorithms for close-in mainbeam jamming scenarios.

3.5 Two-Dimensional Examples of the New Algorithms

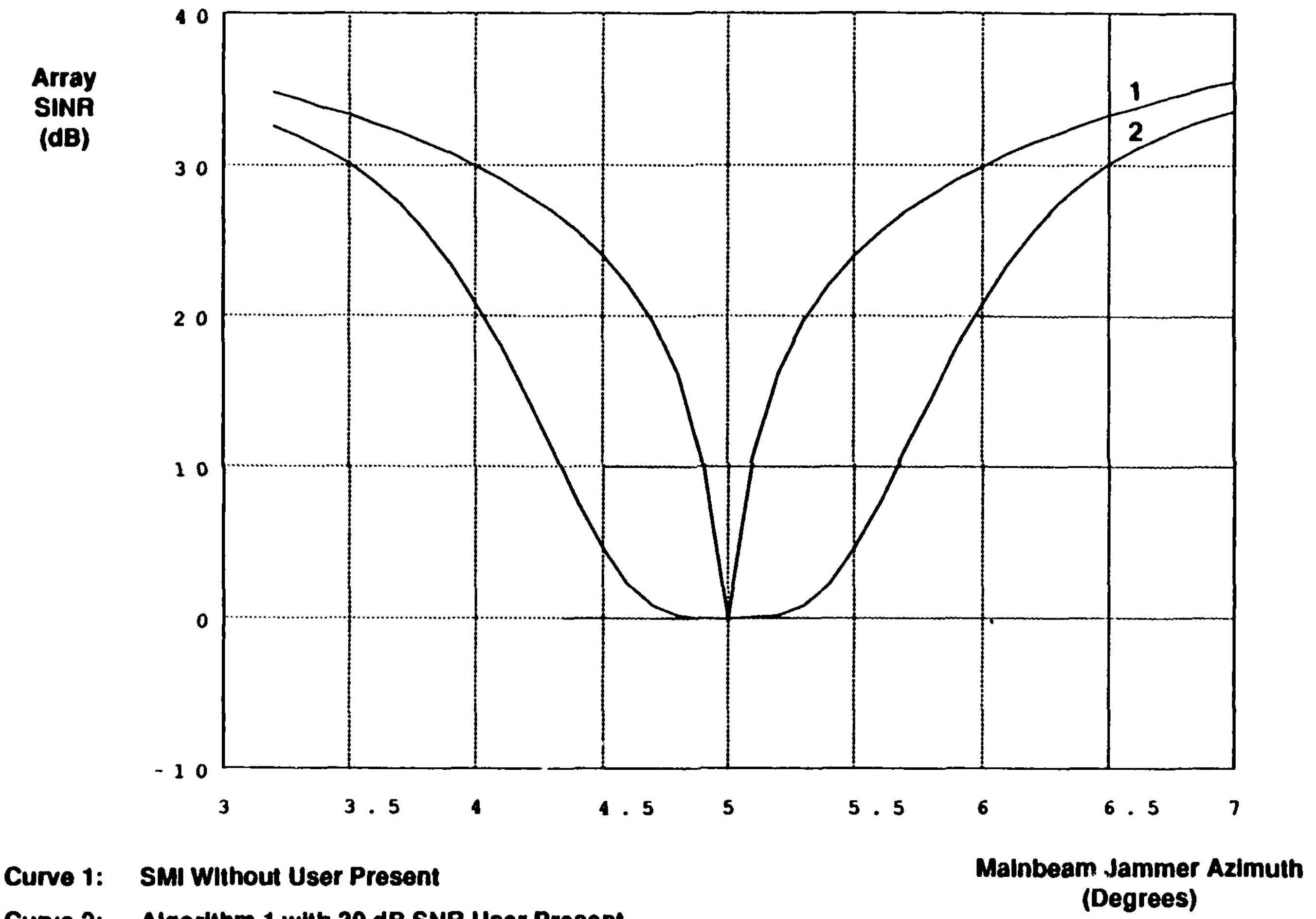
A variety of new nulling algorithms can be generated for two-dimensional arrays by appropriately selecting the weight sets $\vec{w}_{\overline{U}}$ and $\vec{w}_U$, and by specializing the matrix M in (3.35)-(3.52).

One important issue which arises for two-dimensional arrays that is not important for one-dimensional arrays is that of designing the pre-filter with weight set $\vec{w}_{\overline{U}}$. Specifically, nulls in two-dimensional problems occur as contours rather than at discrete points in the one-dimensional problem. For example, Figure 3.21 depicts the antenna pattern for a simple 2×2 array having the weight set given by

$$\vec{w} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{3.62}$$

The corresponding pattern is shown in cosine space; thus the radial coordinate is the cosine of the elevation angle ϕ, where $\phi = 90°$ denotes broadside, and the angular coordinate corresponds to the azimuth θ. The dark loci contours are the loci of points along which the antenna pattern equals zero. Therefore, any emitter located on these contours is nulled by the two-dimensional weight set. Accordingly, *care must be taken in designing the prefilter having weight set $\vec{w}_{\overline{U}}$ so that the contour which nulls the user does not accidentally null*

Figure 3.20: Comparison of Algorithm 1 and SMI as a Function of Mainbeam Jammer Position

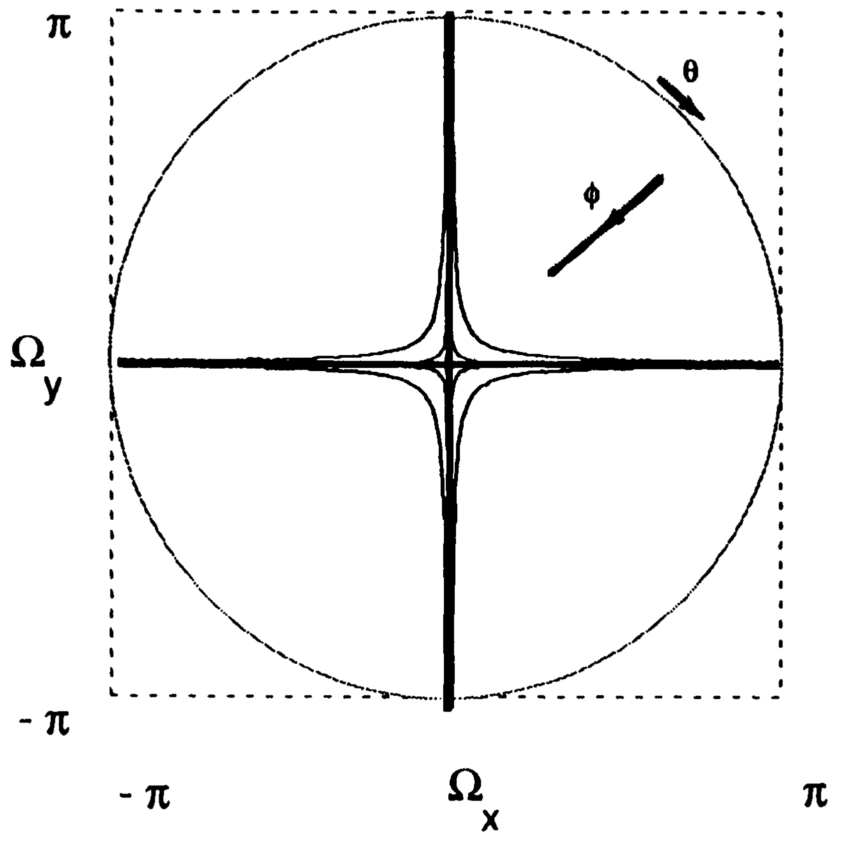


Curve 1: SMI Without User Present

Curve 2: Algorithm 1 with 30 dB SNR User Present

16-Element Linear Array
30 dB JNR Sidelobe Jammer at 50 Degrees
30 dB JNR Mainbeam Jammer at Variable Azimuth

Figure 3.21: Null Contours of Antenna Pattern for a 2x2 Prefilter



Weight	Location
+1	0,0
-1	0,1
+1	1,0
-1	1,1

The only points shown are those with a gain of -30 dB or less.

a jammer. Were this to occur, then the adaptive processor of Figure 3.14 which has the weight set $\vec{w}_J$ would not sense the nulled jammer and therefore would not place a null on the jammer. The composite weight vector $\vec{w} = \vec{w}_U \star \vec{w}_J$ then would pass the jammer.

Assuming that a suitable pre-filter has been selected, then specific nulling algorithms can be generated by specializing the matrix M in equations (3.35)-(3.52). Example algorithms are the following extensions of Algorithms 1-3 of Section 3.4. For purposes of describing the algorithms, we assume that the fixed pre-filter is of size 2×2, and that the composite weight sets $\vec{w}_N$ and $\vec{w}$ are defined on a 6×6 grid. The 2×2 prefilter selected is depicted in Figure 3.22.

Algorithm 1

Select $M = I$ so that $\vec{w}_J = \vec{w}_V$. For the case of a 2×2 prefilter, and a 6×6 array, this results in a weight set $\vec{w}_V$ having $5 \times 5 - 1 = 24$ degrees of freedom.

Algorithm 2

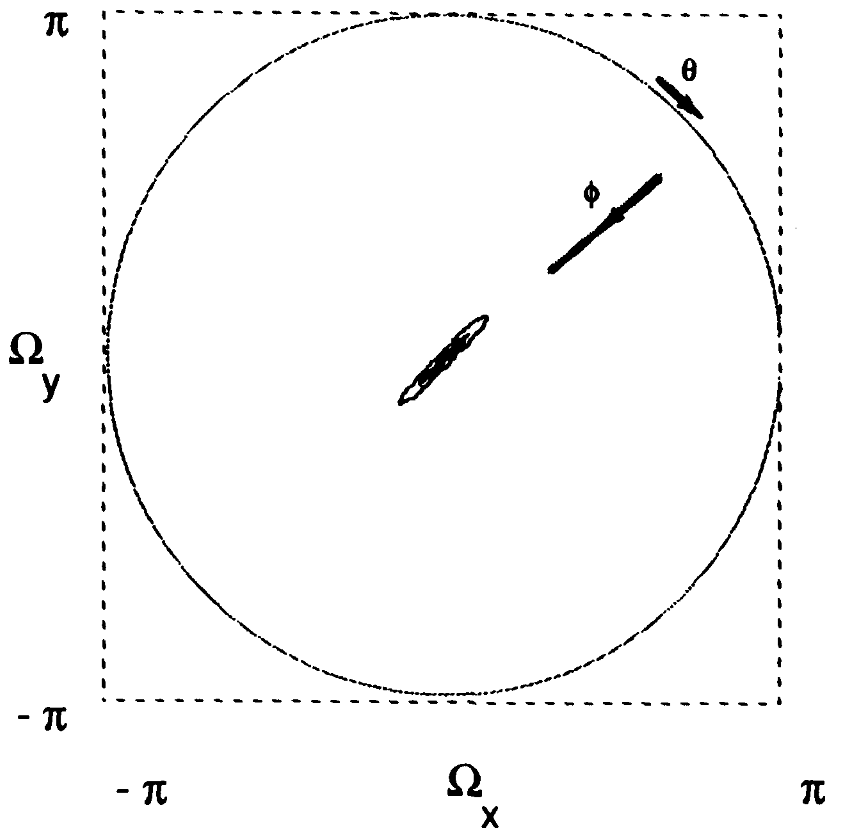
Select M so that its columns correspond to all possible translations of a fixed weight set $\vec{w}_a$ defined on a rectangular $p \times q$ grid. The fixed filter $\vec{w}_a$ is designed to pass the signals from *all* emitters (including the user), and otherwise to have minimum norm. For the 2×2 pre-filter and a 6×6 composite grid, there are $(7-p) \cdot (7-q)$ translations of the weight set $\vec{w}_a$ so that the matrix M is of size $25 \times (7-p) \cdot (7-q)$, and has identical columns except for permutations of elements corresponding to the relative translations of the fixed filter. The vector $\vec{w}_V$ is of length $(7-p) \cdot (7-q)$ and therefore $(7-p) \cdot (7-q) - 1$ degrees of freedom are available.

Algorithm 3

Assume the array is of size $r \times s$. Select M to be of size $(r-1) \cdot (s-1) \times (J+1)$ as described in Algorithm 3 in Section 3.4. Note that the design forces the factorizability of the final weight set and thereby facilitates user restoration.

Based upon a variety of examples that have been considered, we have found that the transient performance of the preceding algorithms is similar to that of their one-dimensional counterparts. Specifically, *the transient response of the algorithms with the user present is comparable to that of SMI with the user absent, and greatly exceeds that of SMI with the user present.*

Figure 3.22: Null Contours of Antenna Pattern for a 2x2 Prefilter



Weight	Location
+1	0,0
-1/3	0,1
-1/3	1,0
-1/3	1,1

The only points shown are those with a gain of -30 dB or less.

The following examples illustrate the performance of Algorithm 1 for three different two-dimensional scenarios. For each example, the pre-filter having weight set $\vec{w}_{\overline{U}}$ is constructed on a 2×2 grid as follows:

- A preliminary set of weights $\vec{w}_{\overline{U}}$ is constructed so that

$$\vec{w}_{\overline{U}} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{3.63}$$

- Next, the projection of the unit-arrival vector $\vec{u}_0$ for the user is removed from the weight set constructed above. The resultant weight set $\vec{w}_{\overline{U}}$ is orthogonal to, and thus nulls, the user.

Figure 3.22 depicts the null contours for a different 2×2 prefilter which was designed to remove the broadside user signal. The contours now extend over a much smaller area as compared with the prefilter in Figure 3.21.

Example 3.9

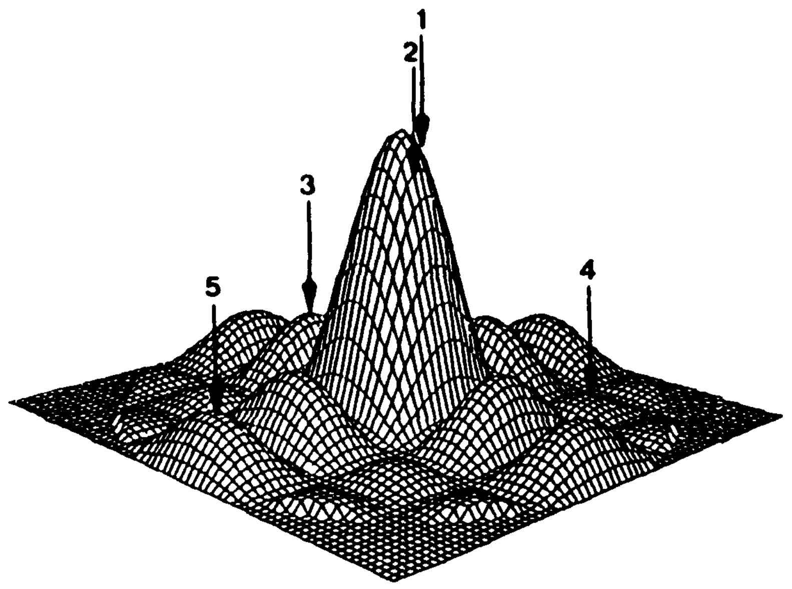
Consider a 6×6 rectangular array of omnidirectional sensors with half wavelength spacing between nearest neighbors. For purposes of describing the two-dimensional scenarios, we use the notation (θ, ϕ) to describe the azimuth (θ) and elevation (ϕ) of each emitter. Assume that a single user is located at $(0°, 90°)$ and that sidelobe jammers are located at $(0°, 60°)$, $(135°, 45°)$, and $(270°, 33.5°)$. Figure 3.23 illustrates the locations of these jammers via the arrows labelled 3,4,5 relative to a uniform-illumination pattern for a 6×6 array steered toward the user.

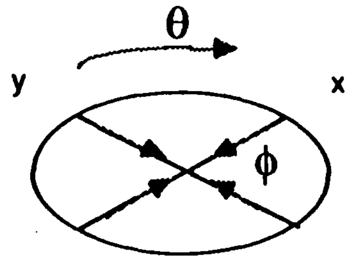
Assume that when the user signal is "on", it transmits a constant envelope signal at a level sufficient to produce a sensor SNR of 30 dB. Assume that the jammers transmit constant envelope signals at levels each adequate to produce a sensor JNR of 30 dB.

Figure 3.24 illustrates the transient response of conventional SMI without and with the user signal present, and Algorithm 1. Curves 1 and 2 depict the array SINR versus N for the user absent and present. Once again, the presence of the user dramatically slows the transient response when the user signal is present in the SMI algorithm. Curve 3 depicts the transient response of the (new) Algorithm 1. Clearly, *the performance of Algorithm 1 with the user signal present equals or exceeds that of SMI with no user, and far exceeds that of SMI with the user.*

End of Example 3.9

Figure 3.23: Location of Jammers Relative to the Antenna Pattern for a 6x6 Weight Set with Uniform Illumination



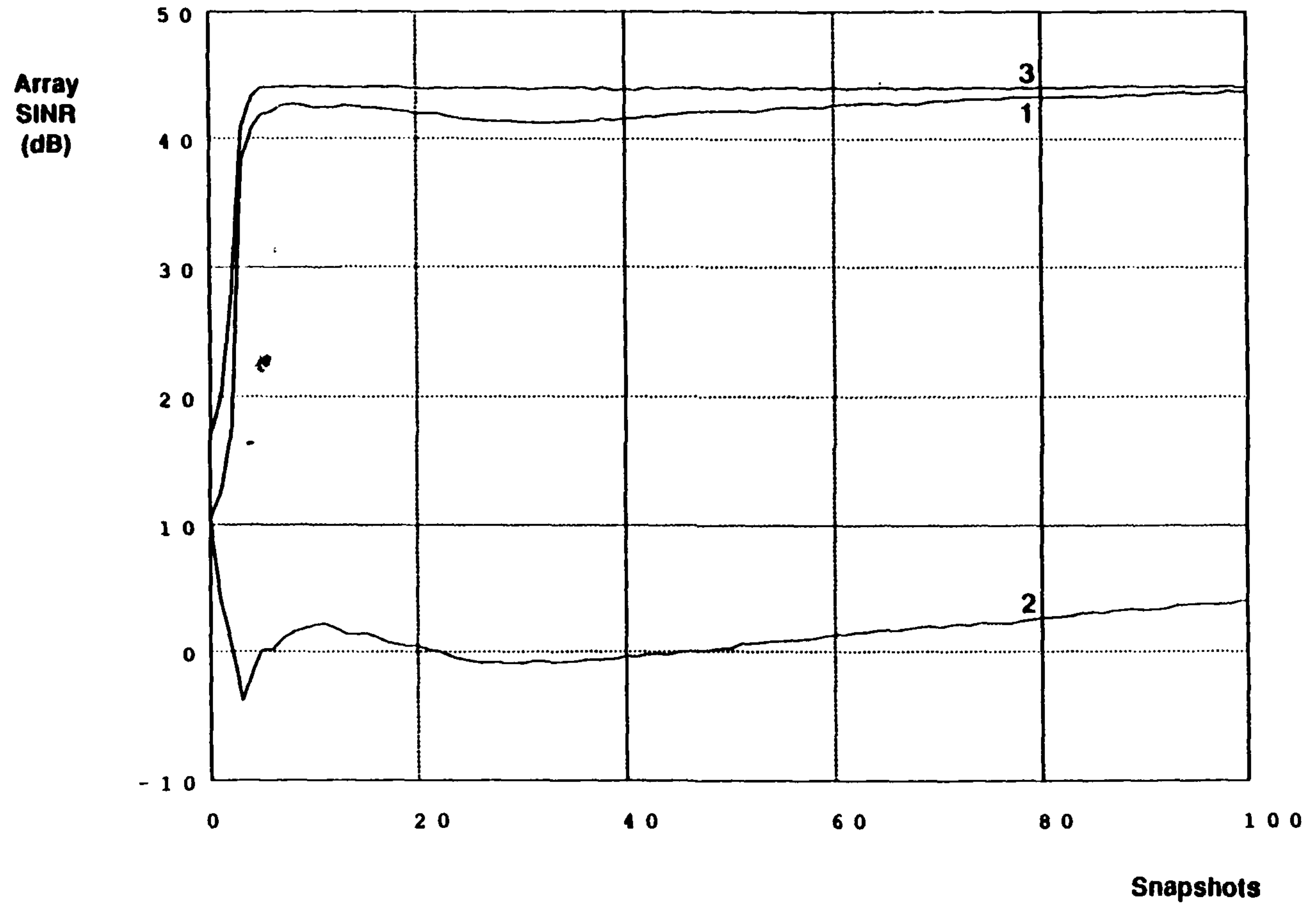


Azimuth = θ
Elevation = ϕ

Jammer Locations

1) θ=90, ϕ=80 (Mainbeam)
2) θ=180, ϕ=85 (Mainbeam)
3) θ=0, ϕ=60 (Sidelobe)
4) θ=135, ϕ=45 (Sidelobe)
5) θ=270, ϕ=33.5 (Sidelobe)

Figure 3.24: Improvement in Planar Array Transient Response Due to Convolutional Filtering (Sidelobe Jamming)



Curve 1: SMI Without User Present
Curve 2: SMI With 30 dB SNR User Present
Curve 3: Algorithm 1 With 30 dB SNR User Present

6x6 Planar Array
30 dB JNR Jammers
at Az/El's of: 0/60, 135/45, 270/33.5

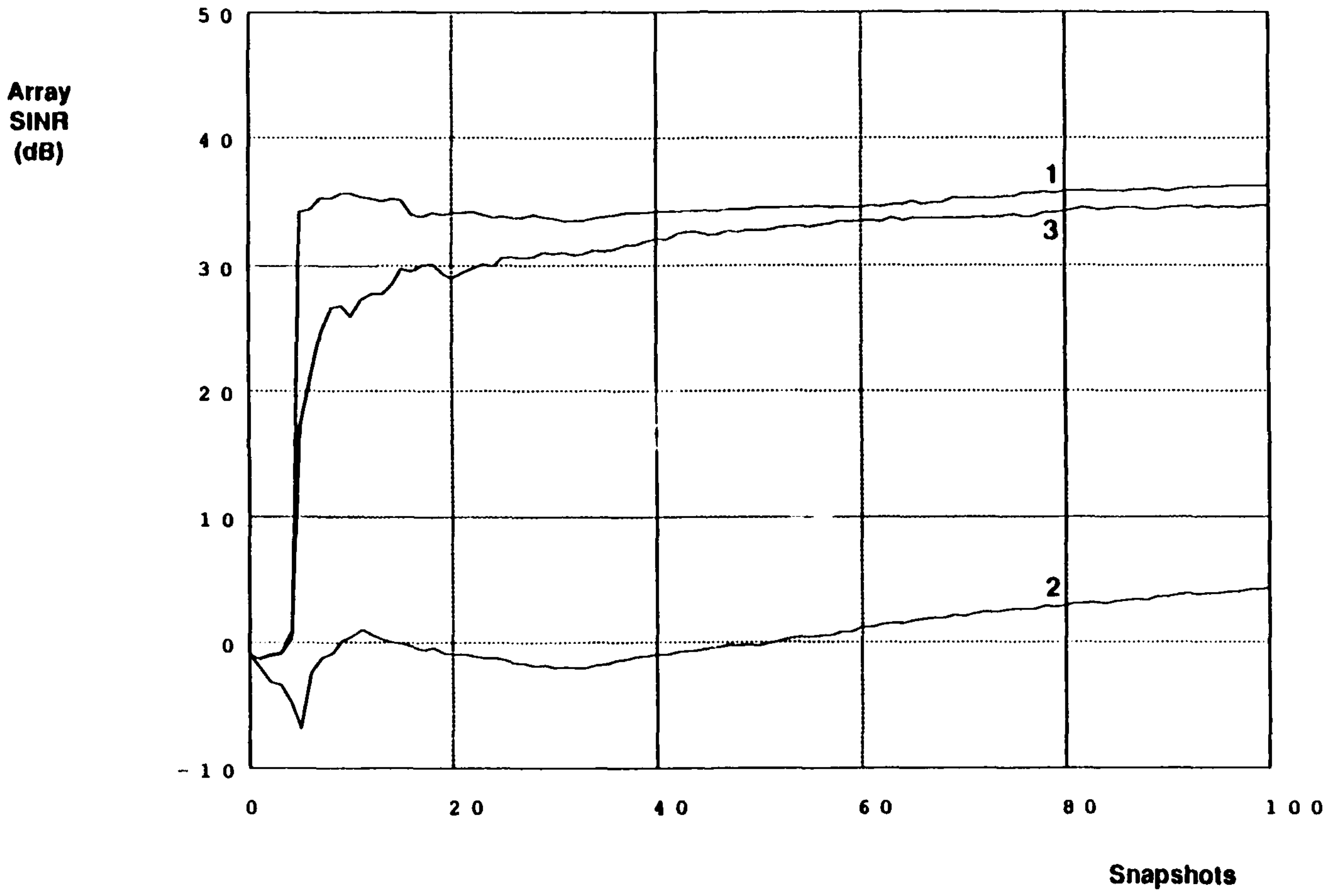
Example 3.10

Two mainbeam jammers were added to the scenario of Example 3.9 to create a combined mainbeam and sidelobe jamming scenario. The mainbeam jammers were placed at the locations $(90°, 80°)$ and $(180°, 85°)$ as illustrated by the arrows labelled as 1 2 and in Figure 3.23,

Figure 3.25 depicts the SINR versus N performance results for SMI without and with the user, and for Algorithm 1 with the user. The performance of Algorithm 1 with the user present equals or exceeds that of SMI with the user present by 30-35 dB for $N \leq 100$.

End of Example 3.10

Figure 3.25: Improvement in Planar Array Transient Response Due to Convolutional Filtering (Mainbeam, Sidelobe Jamming)



Curve 1: SMI Without User Present
Curve 2: SMI With 30 dB SNR User Present
Curve 3: Algorithm 1 With 30 dB SNR User Present

6x6 Planar Array
30 dB JNR Jammers
at Az/El's of: 90/80, 270/33.5,
0/60, 135/45, 270/33.5

Bibliography

[1] S. Applebaum, "Adaptive Arrays", *IEEE Trans. Antennas and Propagat.*, Vol. AP-25, No. 5, pp. 585-598, Sept. 1976

[2] B. Widrow, P. Mantey, L. Griffiths, B. Goode, "Adaptive Antenna Systems", *Proc. IEEE*, Vol. 55, pp. 2143-2159, Dec. 1967

[3] I.S. Reed, J.D. Mallet and L.E. Brennan, "Rapid Convergence Rate in Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-10, No.6, pp. 853-863, Nov. 1974.

[4] B. Widrow, K.M. Duvall, R.P. Gooch, W.C. Newman, "Signal Cancellation Phenomena in Adaptive Arrays: Causes and Cures", *IEEE Trans. Antennas and Propagat.*, Vol. AP-30, No. 3, pp.469-478, May 1982.

[5] O. Frost, "An Algorithm for Linearly Constrained Adaptive Array Processing", *Proc. IEEE*, Vol. 60, pp. 926-935, Aug. 1972.

[6] D.M. Boroson. "Sample Size Considerations for Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-16, pp. 446-451, 1980.

[7] T.W. Miller, "The Transient Response of Adaptive Arrays in TDMA Systems", Ph.D. Dissertation, Department of Electrical Engineering, The Ohio State University, 1976.

[8] R.A. Monzingo and T.W. Miller, *Introduction to Adaptive Arrays*, Chap. 6, John Wiley and Sons, New York, 1980.

[9] "Advanced Nulling Techniques: Phase I Final Report", Atlantic Aerospace Electronics Corporation (Formerly Pollard Road Inc.), AF Contract Number F04701-85-C-0078, Mar. 1986.

[10] L.J. Griffiths and C.W.Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming", *IEEE Trans. Antennas and Propagat.*, Vol. AP-30, No. 1, pp. 27-34, Jan. 1982.

[11] "Spatial/Spectral Filtering with Linearly Constrained Minimum Variance Beamformers", *IEEE Trans. Acoust., Speech and Sig. Proc.*, Vol. ASSP-35, No. 3, pp. 249-266, Mar. 1987.

[12] W.F. Gabriel, "Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems", *IEEE Trans. Antennas and Propagat.*, Vol. AP-34, No. 3, pp. 291-300, Mar. 1986.

Chapter 4

Broadband Techniques

Contents

List of Figures

4.1 Introduction

This chapter examines alternative nulling techniques applicable to broadband communications problems. We assume that the receiving array is illuminated by signals from one broadband user in a ***known direction***, and also by broadband jammers in ***unknown directions.*** The objective is to extract, by spatial filtering, a desired broadband user signal from a broadband jamming environment, and to do so ***without significantly distorting the user signal.***

We examine the following techniques.

Narrowband Arrays (i.e., weight-and-sum processors):

1. Conventional nulling techniques which select the weights by minimizing the array output power subject to one or more user-gain constraints.
2. A new class of techniques which automatically place ***multiple nulls*** on (or near) jammers rather than simple nulls. These techniques extend both the spectral and spatial ranges of potential suppression well beyond those actually occupied by the jammer signals.
3. A variant of the foregoing class of techniques which we designate as ***elevation null techniques.*** [1] Elevation nulls achieve the same objectives as multiple nulls, but with a reduction in the number of degrees of freedom consumed.

Broadband Arrays (i.e., filter-and-sum processors):

1. Conventional nulling techniques which select weights by minimizing the output power subject to one or more user-gain constraints.
2. A new class of techniques which places multiple nulls on (or near) jammers rather than simple nulls.

Since issues relating to the transient behavior of nulling systems have been explored in Chapter 3, we limit consideration here to the steady state nulling properties of the techniques.

[1] In previous working papers we have also referred to elevation nulls as radial nulls. In the two-dimensional sine-space representation of the beam pattern, azimuth refers to polar angle, while elevation refers to the radial distance. Thus, elevation nulls are equivalent to radial nulls.

Our main conclusions are as follows:

Narrowband Arrays

1. Power minimization subject to a single-frequency gain constraint is *not* an effective method for selecting weights given a broadband user at an off- broadside angle. Specifically, the resultant weight set not only nulls the jammers, but also nulls components of the user signal thereby producing distortion. In effect, the algorithm treats the user signal components removed from the constraint frequency as mainbeam jammers, and attempts to suppress these components. To be effective, power minimization techniques must incorporate *multiple* user-gain constraints to assure consistency of user gain across the entire (instantaneous) user frequency band.

2. Weights produced by constrained power-minimization to suppress a broadband jammer provide suppression over an angular sector rather than in a single direction as in the narrowband problem. (In one-dimensional problems, the sector is in the azimuth coordinate measured relative to the array boresight; in two-dimensional problems, the sector is in the elevation coordinate measured relative to the plane of the array.) The sectoral null in the pattern assures that the jammer is suppressed as the antenna pattern scans in angle in response to different jammer frequency components. Due to the sectoral suppression of jammers, these techniques are not well-suited for suppressing a jammer close to the user since the techniques tend to suppress not only the jammer, but also the user.

3. Multiple nulls potentially are useful in avoiding the nulling system problems that result from the presence of a user signal (see Chapters 3 and 5). Specifically, the potential exists to filter the user signal out of the sensor data using band-pass filters, and to then determine the nulling weights based upon the jamming signals in the adjacent frequency bands. Since multiple nulls provide spectral extension of nulls, the resultant weights also suppress the in-band jamming signals. Multiple nulls also potentially are useful in problems in which the array is mounted on a platform that can significantly change its angular orientation over the time interval required to calculate a new set of nulling weights.

4. Elevation nulls achieve the same spectral benefits as multiple nulls while consuming fewer degrees of freedom. However, computation of weights to produce the nulls is more involved than that for multiple nulls.

5. Due to extended spatial depression of the antenna pattern, multiple-null and elevation-null techniques are less well-suited than constrained power-minimization techniques for suppressing a jammer close to the user.

<u>Broadband Arrays</u>

1. As is the case with narrowband arrays, multiple user-gain constraints must be employed with power-minimization techniques in order to suppress jammers, and to pass broadband user signals with acceptable distortion.

2. In contrast to narrowband arrays, broadband arrays produce sharp spatial nulls on jammers; that is, the nulls are of very limited extent both in azimuth and in elevation. Thus, broadband arrays are better suited than narrowband arrays for suppressing broadband jammers that are located close to the user.

3. Multiple nulls in broadband arrays potentially are useful in the same ways as for narrowband arrays; that is, to avoid problems associated with presence of the user signal, and also to reduce the impact of platform motion.

4. Owing to the large number of degrees of freedom, broadband arrays can require substantial computation to determine nulling weights. Thus, it is highly desirable to employ beamformers with thinned beam sets to reduce the required computation (see Section 2.5.4)

This chapter is organized as follows. Section 4.2 addresses the issues and opportunities associated with using narrowband arrays to extract a broadband user signal from a broadband jamming environment. This section includes a discussion and examples of linear constraints, multiple nulls and elevation nulls. Section 4.3 addresses issues and characteristics associated with broadband arrays for the similar problem. This section includes a discussion and examples of linear constraints and antenna patterns as extensions of the algorithms to incorporate multiple nulls.

4.2 Narrowband Arrays

4.2.1 Equivalence Of Spatially Discrete Broadband Emitters And Spatially Distributed Narrowband Emitters

Assume that an emitter is transmitting a plane wave signal toward a uniform *linear array* with K sensors from an off-broadside angle of θ_1. For a (narrowband) signal at angular frequency ω, the array snapshot at the n^{th} sampling instant is of the form

$$\vec{x}(n) = a(n)\,[1, e^{j(d/c)\omega\sin\theta_1}, \ldots e^{j(K-1)(d/c)\omega\sin\theta_1}]^T \tag{4.1}$$

where d denotes the sensor separation, c denotes the speed of light, and $a(n)$ denotes the complex amplitude of the wave. Note that the snapshot $\vec{x}(n)$ is identical to the snapshot

$$\vec{x}(n) = a(n)[1, e^{j(d/c)\omega_1\sin\theta}, \ldots e^{j(K-1)(d/c)\omega_1\sin\theta}]^T \tag{4.2}$$

for frequency ω_1 provided the angle θ satisfies the relationship

$$\omega_1 \sin\theta = \omega\sin\theta_1 \tag{4.3}$$

or

$$\theta = \sin^{-1}[(\omega/\omega_1)\sin\theta_1] \tag{4.4}$$

If $\omega_1 < \omega$, then the solution of (4.4) satisfies

$$\theta > \theta_1 \tag{4.5}$$

Thus an emitter transmitting at frequency ω from angle θ_1 induces exactly the same (sampled) signals in the array as an emitter transmitting at the lower frequency ω_1 from the larger angle θ.

This circumstance explains a fundamental property of narrowband linear arrays used in broadband contexts [1][2][3][4][5]. Specifically, a single snapshot for a broadband emitter having frequencies in the band

$$\omega_1 \le \omega \le \omega_2 \tag{4.6}$$

transmitting from the angle θ_1 is identical to the snapshot for a sum of signals at frequency ω_1 transmitted from the continuum of angles

$$\theta_1 \le \theta \le \sin^{-1}[(\omega_2/\omega_1)\sin\theta_1] \tag{4.7}$$

More generally, if J broadband jammers in the frequency band (4.6) transmit signals that are incident upon the array from discrete angles θ_j $j = 1 \cdot \cdot J$, then a single array snapshot for the J jammers is identical to a snapshot for J families of narrowband jammers at frequency ω_1 transmitting from the J sectors defined by the relationships

$$\theta_j \leq \theta \leq \sin^{-1}[(\omega_2/\omega_1)\sin\theta_j] \tag{4.8}$$

Consequently, a weight-and-sum processor operating on the linear array processes the J broadband jamming signals exactly as would it process J families of narrowband jamming signals at frequency ω_1 arriving from the angular sectors (4.8). As a result, the processor for the linear array *allocates the available degrees of freedom so as to null out angular sectors* rather than individual angles; moreover the wider the bandwidth, the broader the sector.

A corresponding result can be developed for a planar array of sensors. Specifically with regard to the coordinate system shown in Figure 4.1, the snapshot for a plane wave at frequency ω arriving from the direction (θ_j, ϕ_j) is identical to one for a plane wave at frequency ω_1 from the direction (θ_j, ϕ) where ϕ satisfies the equation

$$\omega_1 \cos\phi = \omega \cos\phi_j \tag{4.9}$$

or

$$\phi = \cos^{-1}[(\omega/\omega_1)\cos\phi_j] \tag{4.10}$$

Note that if $\omega > \omega_1$, then the solution ϕ of (4.10) satisfies

$$\phi < \phi_j \tag{4.11}$$

It follows that J broadband jammers occupying the frequency band (4.6), and transmitting from directions (θ_j, ϕ_j), for $j = 1 \cdot \cdot J$, induce exactly the same snapshots in the array as J families of narrowband jammers transmitting at frequency ω_1 from discrete azimuths θ_j, and from continuous elevation sectors defined by

$$\cos^{-1}[(\omega_2/\omega_1)\cos\phi_j] \leq \phi \leq \phi_j \tag{4.12}$$

As a result, a weight-and-sum processor operating on the snapshots for a planar array allocates the available degrees of freedom so as to null out the *angular regions* defined by the relationships:

1. $\theta = \theta_j$,

2. ϕ satisfying (4.12)

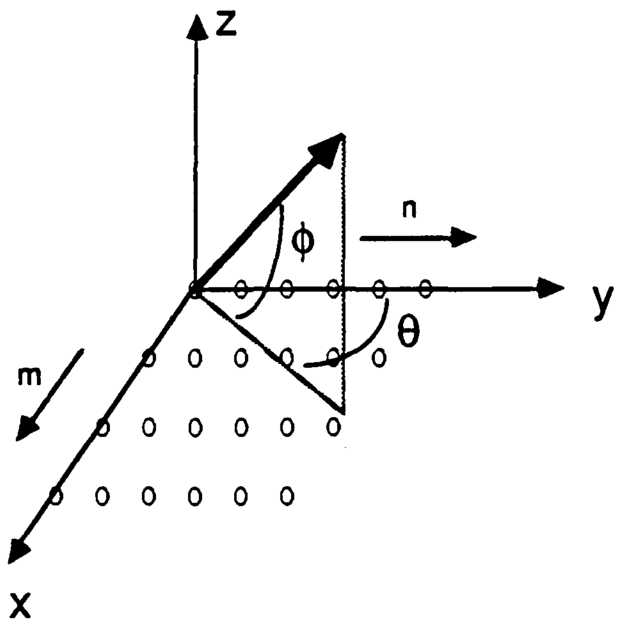


Figure 4.1: 2D Array geometry.

M sensors are positioned along the x-axis and N sensors are positioned along the y-axis. Azimuth is measured with respect to the y-axis and elevation is measured with respect to the x,y plane.

for $j = 1 \cdot\cdot J$. That is, *the processor suppresses signals arriving from the discrete azimuth θ_j and the elevation sector (4.12) for $j = 1 \cdot\cdot J$.*

The following examples illustrate this point.

Example 4.1: Spatial and Spectral Properties of Nulls Produced by a Linear Array

Consider a uniform linear array of 16 omni-directional sensors. Assume an interference environment which consists of a 0 dB monochromatic jammer with a normalized frequency of 1.0, located at an azimuth of $-30°$, and a 0 dB fractional bandwidth jammer, extending over the normalized frequency band from 0.75 to 1.0, located at an azimuth of $+30°$. [2] Assume that each jammer produces a 40 dB JNR. Also assume that no user signal is present.

The Wiener-Hopf equation was solved for a set of nulling weights assuming an omni-directional quiescent antenna pattern (or steering vector). With no jammers present, the beampattern of the array would be flat over all azimuths, since the quiescent pattern is omni-directional. With jammers present however, the pattern is expected to exhibit nulls at the jammer location of $\pm 30°$.

Figure 4.2 shows the beampattern for the weight set at a normalized frequency of 1.0, plotted as a function of azimuth. From the figure, it can be seen that both jammers are effectively suppressed - the depth of the null at $-30°$ is much deeper than -60 dB, while the gain of the array in the $+30°$ look direction is down by more than 45 dB. Note the difference in the spatial extent of the two nulls however. The null in the direction of the narrowband jammer is quite sharp in spatial extent, while the null in the direction of the 25% fractional bandwidth jammer is much broader. Also, the gain of the array is down by at least 20 dB in a 12° region about the fractional bandwidth jammer. In some applications, this effect may be undesirable, since gain to narrowband users which are also within an angular region of the fractional bandwidth jammer is reduced.

Figure 4.3 shows the frequency response of the array in a $+30°$ look direction, i.e., in the direction of the fractional bandwidth jammer, for the same set of weights. Note that the gain of the array is down by at least 45 dB across the fractional bandwidth from 0.75 to 1.0. Reference to Figure 4.2 shows that the gain variations as frequency is increased from 0.75 to 1.0 are virtually identical to the gain variations as azimuth is increased from 20° to 30°

[2] For examples in this chapter, frequencies are normalized such that a normalized frequency of 1.0 corresponds to the frequency at which sensors are spatially separated by a half-wavelength.

Figure 4.2: Beampattern (at f = 1.0) for a Linear Array (Example 4.1)

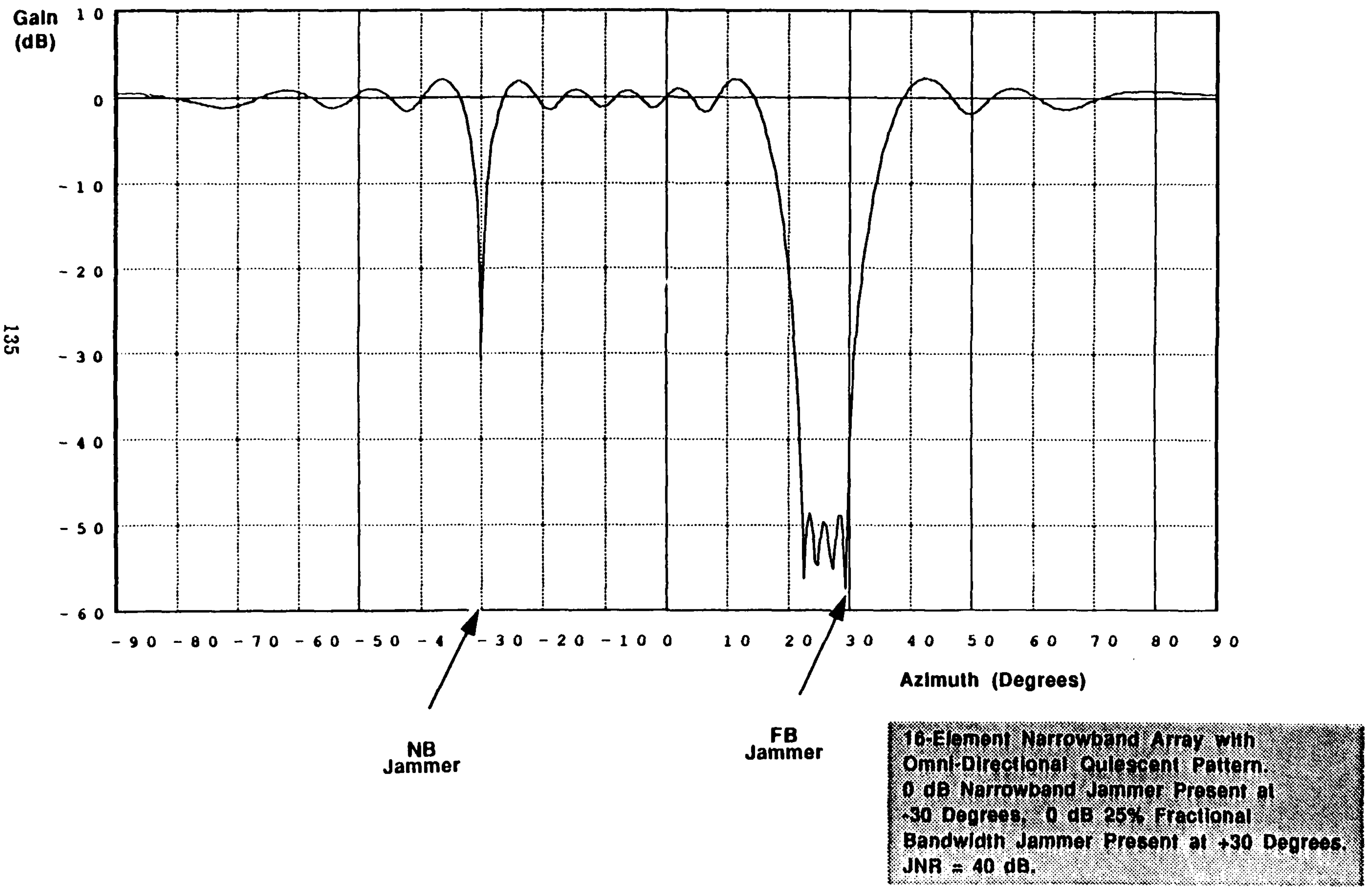
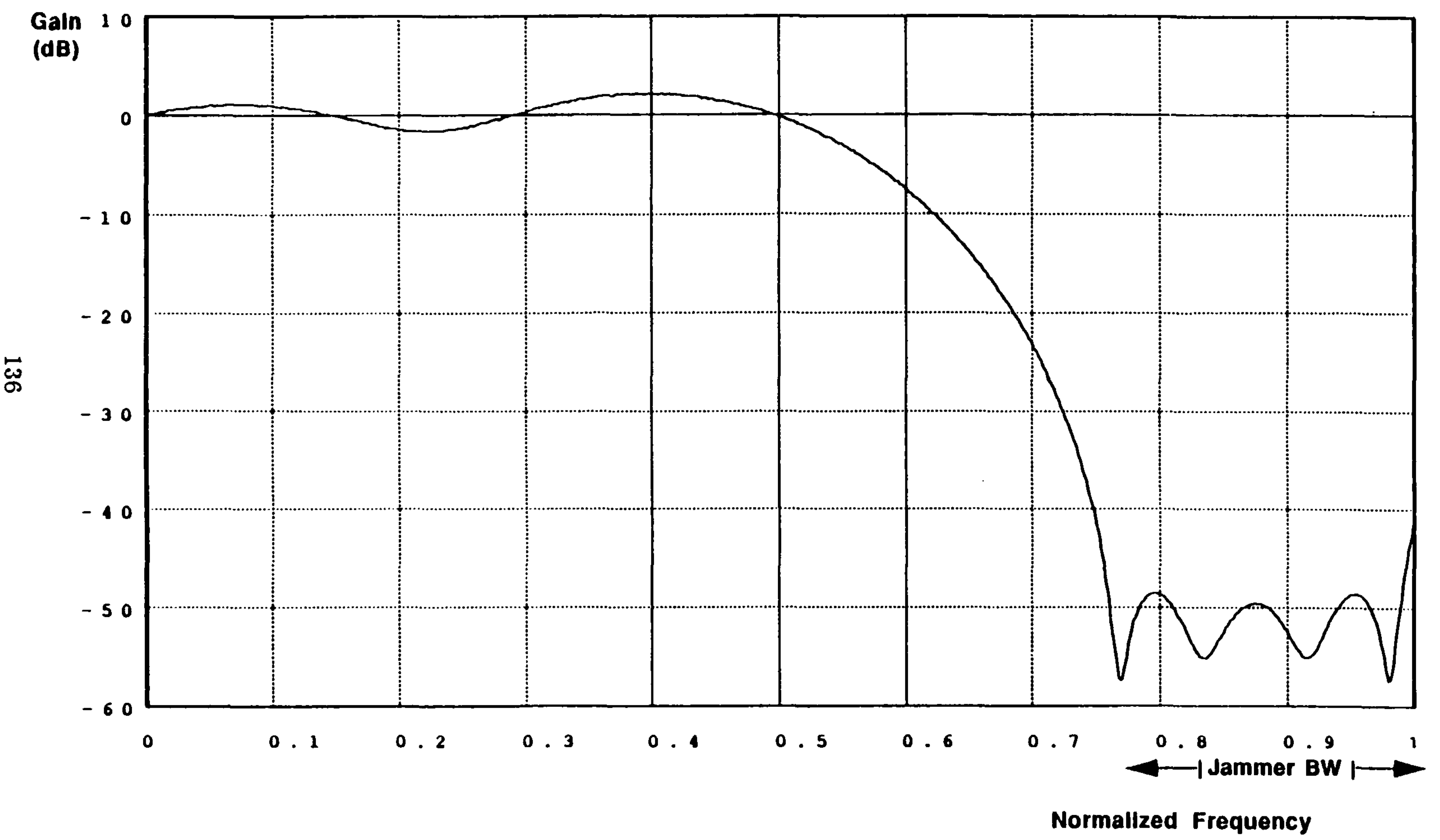


Figure 4.3: Frequency Response of a Linear Array in the Look Direction of the 25% Fractional Bandwidth Jammer at 30 Degrees (Example 4.1)



16-Element Narrowband Array with Omni-Directional Quiescent Pattern. 0 dB Narrowband Jammer Present at -30 Degrees. 0 dB 25% Fractional Bandwidth Jammer Present at +30 Degrees. JNR = 40 dB.

End of Example 4.1

Example 4.2: Spatial and Spectral Properties of Nulls Produced by a Planar Array

Consider a 6x6 planar array. To illustrate spatial-frequency relationships, an interference environment was chosen to consist of three jammers with identical powers, with differing fractional bandwidths and locations. The characteristics of the three jammers are summarized in Table 4.1. The JNR for each jammer is ratio of the jammer power density

Jammer	Azimuth	Elevation	Power (dB)	Norm. Bandwidth
A	30	60	0	1.0-1.0
B	140	60	0	0.95-1.0
C	280	60	0	0.7-1.0

Table 4.1: Jammer Characteristics for Example 4.2

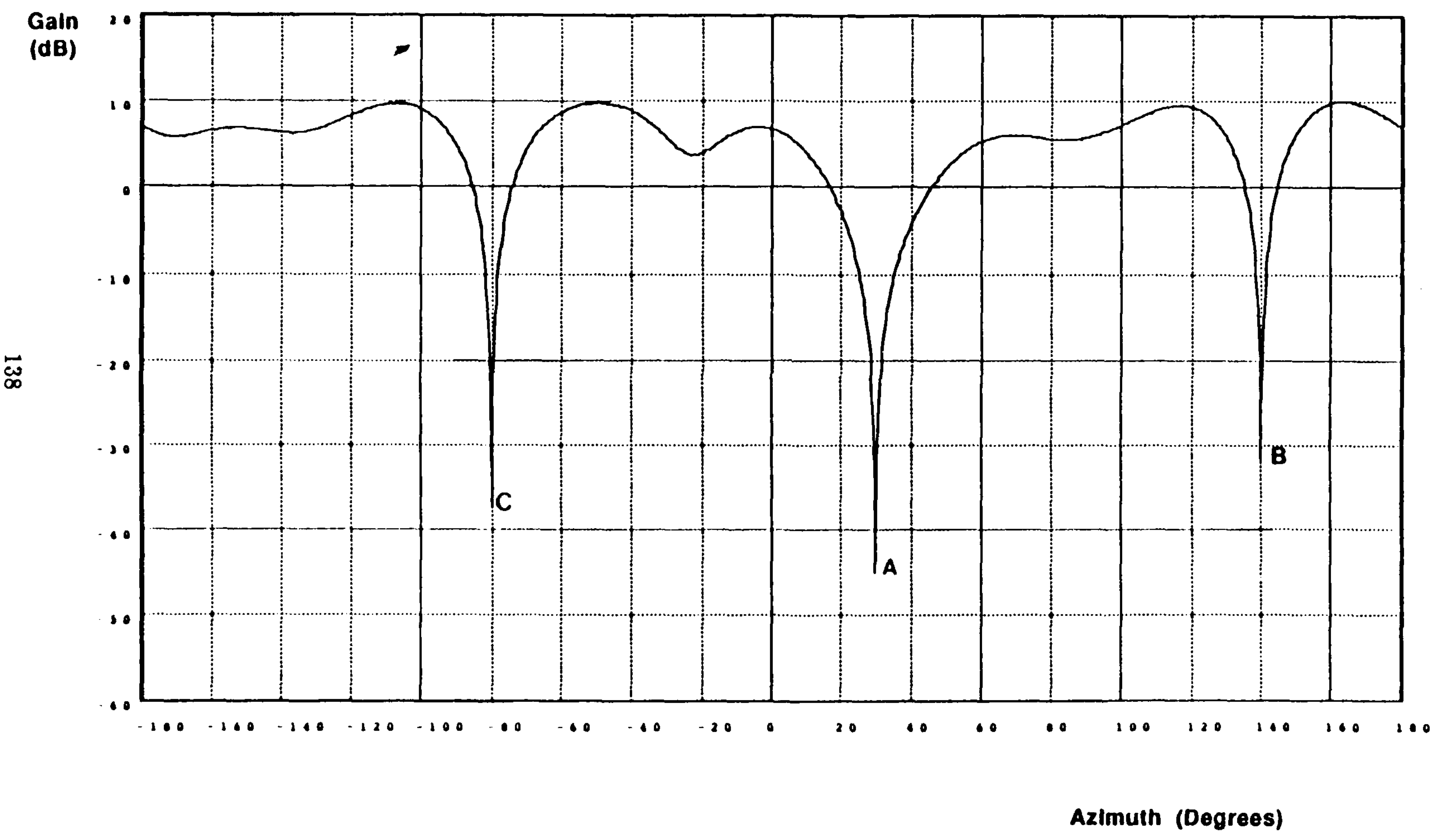
integrated across the relevant frequency band to the total noise power. A jammer-to-noise ratio (JNR) of 40 dB was chosen for each of the three jammers.

The sensor covariance matrix was constructed for the foregoing interference environment, and the Wiener equation was solved assuming a quiescent weight vector corresponding to an omni-directional antenna pattern. Under the conditions of no jammers present, the beampattern of the array would be flat at all azimuth and elevation angles, since the quiescent pattern is omni-directional. With jammers present however, the pattern is expected to exhibit nulls at the jammer locations.

Figure 4.4 shows the resultant beampattern at the normalized frequency of 1.0, plotted as a function of azimuth for a fixed elevation of 60 degrees, i.e. at the common elevation angle of the three jammers. Note that the nulls are sharp functions of azimuth for all three jammers. Figure 4.5 depicts the beampattern at a normalized frequency of 1.0, as a function of elevation for the three jammer azimuths. Note that while the nulls on the three jammers are quite sharp in azimuth (see Figure 4.4), the extent of a null in elevation widens as the fractional bandwidth of a jammer increases (see Figure 4.5).

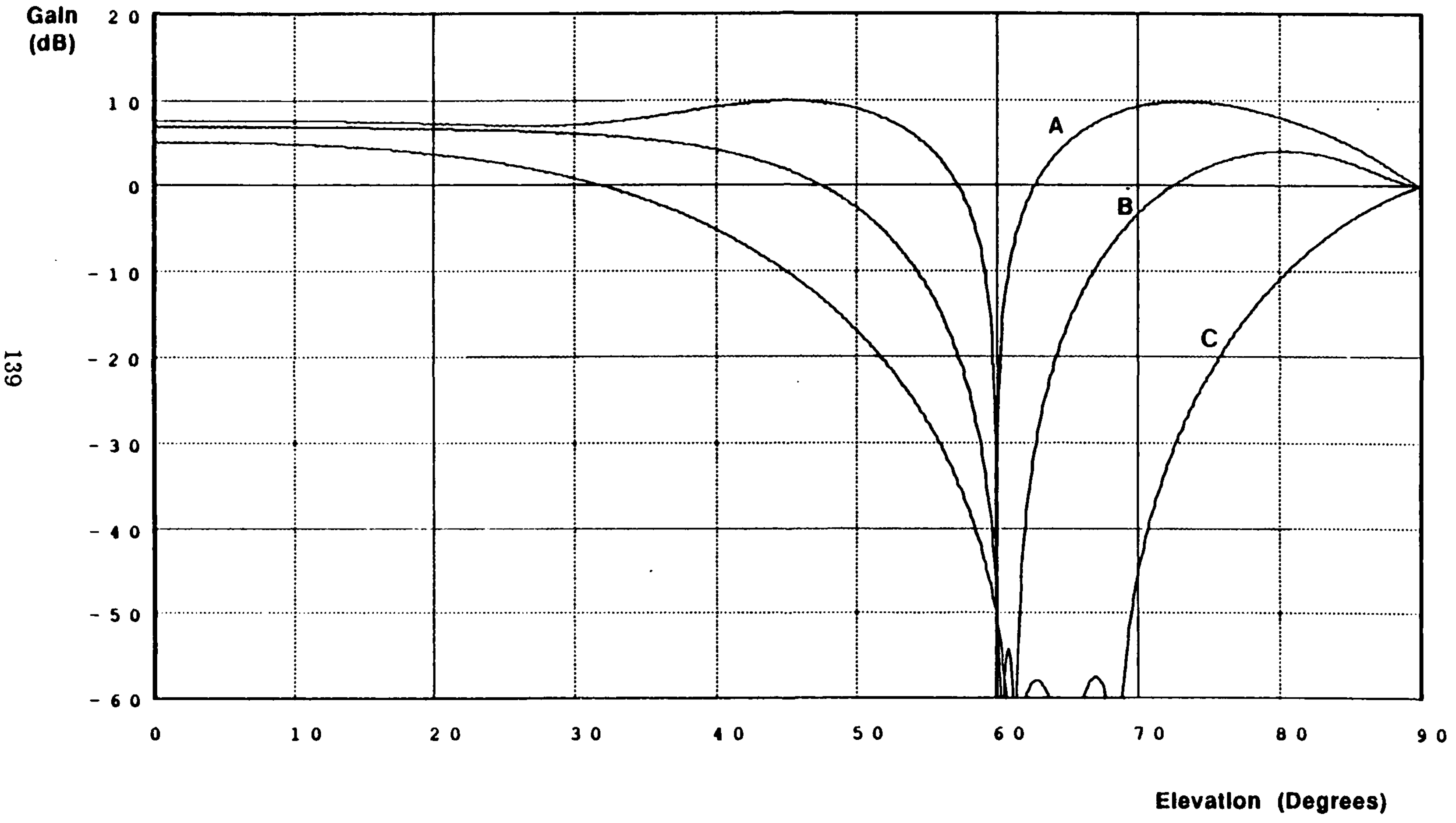
Figure 4.6 shows the frequency response of the array in the look directions of the three jammers. Note that the array provides effective nulling over the bandwidths of the three jammers. However, Figures 4.4,4.5, and 4.6 in conjunction show that, while effective nulling in bandwidth is provided by the optimal planar array, suppression of a user signal also occurs

Figure 4.4: Beampattern (at f = 1.0) for a Planar Array. Elevation = 60° . (Example 4.2)



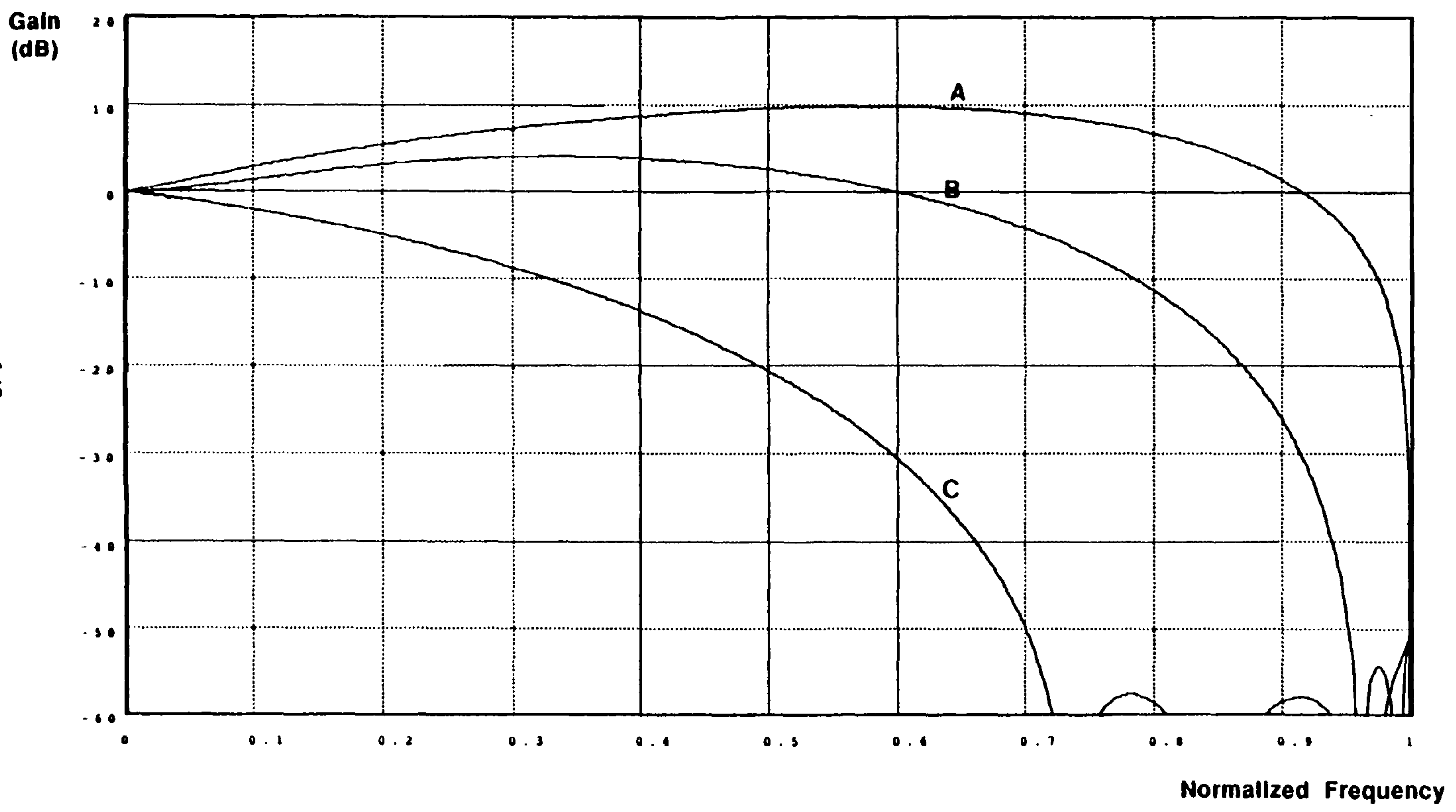
6x6 2D Narrowband Array With Omni-Directional Quiescent Pattern. Azimuth/Elevation of the 3 Jammers Was 30/60(A), 140/60(B), and 280/60(C). Fractional Bandwidth of Jammers Was 0%(A), 5%(B), 30%(C). All Jammers 40 dB JNR.

Figure 4.5: Beampattern (at f = 1.0) for a Planar Array at Azimuths of 30°, 140°, 280° (Example 4.2)



6x6 2D Narrowband Array With Omni-Directional Quiescent Pattern. Azimuth/Elevation of the 3 Jammers Was 30/60(A), 140/60(B), and 280/60(C). Fractional Bandwidth of Jammers Was 0%(A), 5%(B), 30%(C). All Jammers 40 dB JNR.

Figure 4.6: Frequency Response of a Planar Array in the Look Direction of Jammers A, B, C (Example 4.2)



6x6 2D Narrowband Array With Omni-Directional Quiescent Pattern. Azimuth/Elevation of the 3 Jammers Was 30/60(A), 140/60(B), and 280/60(C). Fractional Bandwidth of Jammers Was 0%(A), 5%(B), 30%(C). All Jammers 40 dB JNR.

over a relatively large range of elevation angles.

End of Example 4.2

4.2.2 Power Minimization with Multiple Gain Constraints

When one utilizes a narrowband array in a broadband context, the objective typically is to spatially filter the array snapshots so as to pass one broadband user signal without distortion while suppressing broadband jamming signals. An obvious method for pursuing this objective is to select the array weights as in the narrowband problem. That is, one selects the weights so as to minimize the array output power subject to a single user gain constraint at a selected frequency ω_0. This approach leads to solving then classical Wiener-Hopf equation for the weight set. The purpose of this section is to show that ***constrained power minimization fails to provide effective array weights in broadband applications when only a single user gain constraint is employed*** - the resultant weight set typically distorts the user signal. However, ***if one utilizes multiple gain constraints to assure constant gain and group delay across the full user frequency band, then constrained power minimization continues to be a viable method by which to select weights.*** This approach leads to using the weight set (2.38) in place of (2.34)[6][7][8]

Thus, assume that a uniform linear array with a weight-and-sum processor is illuminated by a single broadband user transmitting from an off-broadside angle of θ_0, and also by multiple broadband jammers transmitting from other directions. Consider the consequences of selecting the array weights so as to minimize the array output power subject to a single constraint that requires gain in the user direction θ_0 to equal a prescribed value (e.g., unity) at a selected frequency ω_0. The results of the preceding section show that a broadband emitter in a discrete spatial direction appears exactly the same to the (sampled) array as a single frequency ω_0 emitter distributed across an angular sector. Therefore, the single gain constraint insures only that gain is maintained in the direction θ_0 at frequency ω_0.

Specifically, there is no requirement that gain be maintained in the other apparent user directions at ω_0. Thus, the ***subsequent power-minimization step treats the user signal components apparently arriving from directions other than θ_0 as interference.*** Consequently, the processor allocates the available degrees of freedom so as to suppress both the apparent interference signals, and also the actual jammer signals. Thus, the calculated weight set (2.34) passes the user signal components apparently arriving from the direction θ_0, but suppresses the user signal components apparently arriving from other directions. As a result, ***the weight set (2.34) distorts the user signal.*** The following example illustrates this point.

Example 4.3: Performance of a Narrowband (Weight-and-Sum) Adaptive Processor with a Single Constraint in the Presence of Broadband Jammers and Broadband User [Linear Array]

Consider a 16-element uniform linear array which is illuminated by broadband user and jammer signals. The user and jammer characteristics are described in the following table. The SNR (JNR) for each emitter refers to integrated power across the corresponding frequency band.

Signal	Angle	Freq. Band	SNR,JNR (dB)
User	+30°	0.9 – 1.0	40
Jammer 1	+65°	0.9 – 1.0	40
Jammer 2	−10°	0.9 – 1.0	40

Table 4.2: Parameters for the One-Dimensional Broadband Scenario for Examples 4.3 and 4.4

Assume that the weight vector is selected by power minimization subject to a single constraint of unity gain at the lower edge of the user frequency band, i.e. at $f = 0.9$. Figure 4.7 depicts the antenna pattern for the resultant weight vector at the frequency $f = 0.9$. Note that there is sectoral suppression of the jammers, due to their broadband character.

Note also the strong suppression of signals to the right side of the user direction $\theta_0 = 30°$. This circumstance is a result of the array treating the higher frequency user signals as mainbeam jamming signals arriving from angles greater than 30°, and utilizing the available degrees of freedom so as to suppress these user signal components.

Figure 4.8 depicts the frequency response in the actual direction of the user. Clearly, the array provides the prescribed unity gain at the frequency $f = 0.9$. However, the gain falls off sharply at the higher user frequencies. Therefore, ***the array severely distorts the user signal due to the removal of these components.***

End of Example 4.3

Figure 4.7: Beampattern (at f = 0.9) for a Linear Array After Adapting in the Presence of a Broadband User and Two Broadband Jammers. Single Gain Constraint (Example 4.3)

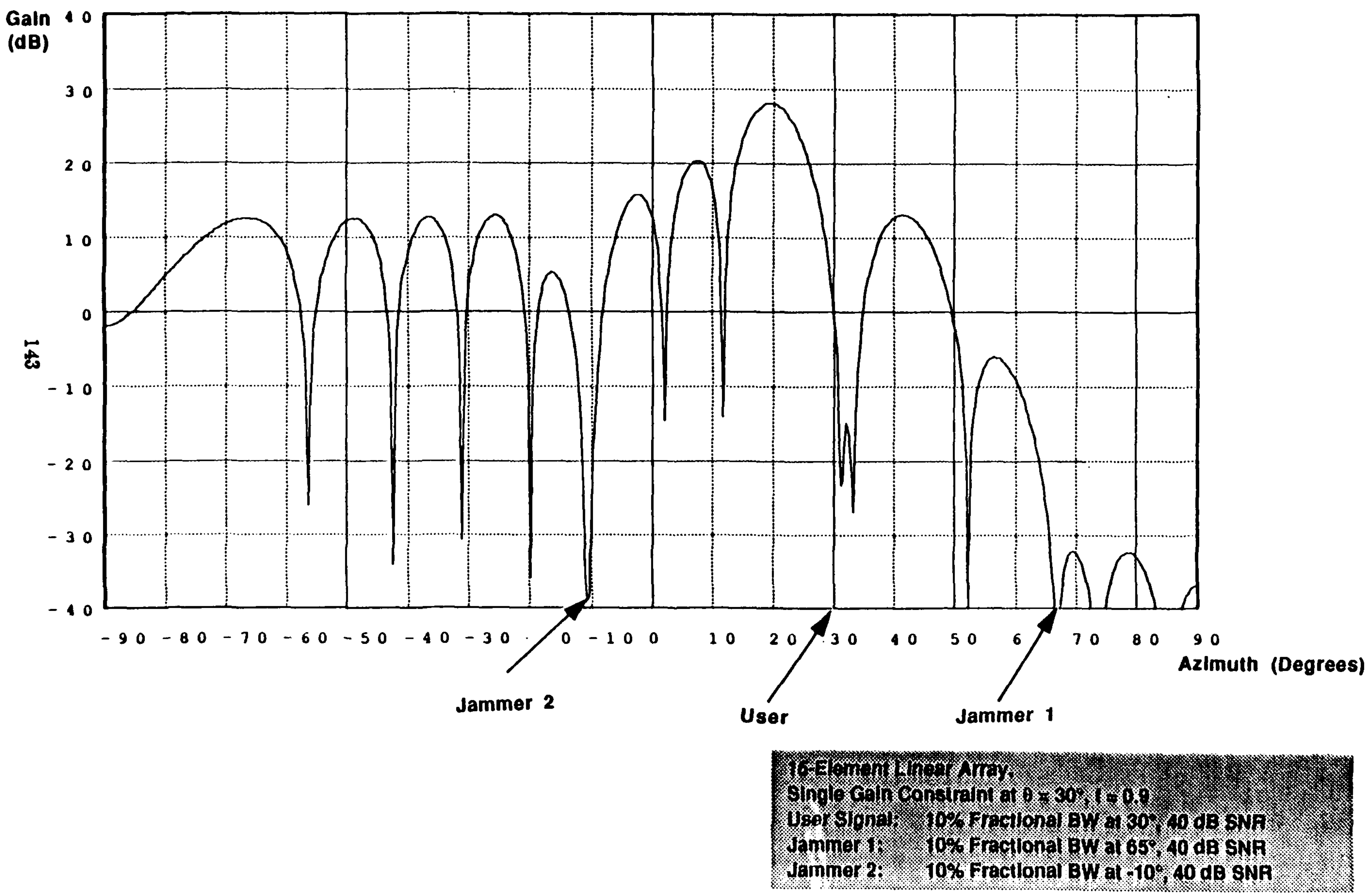
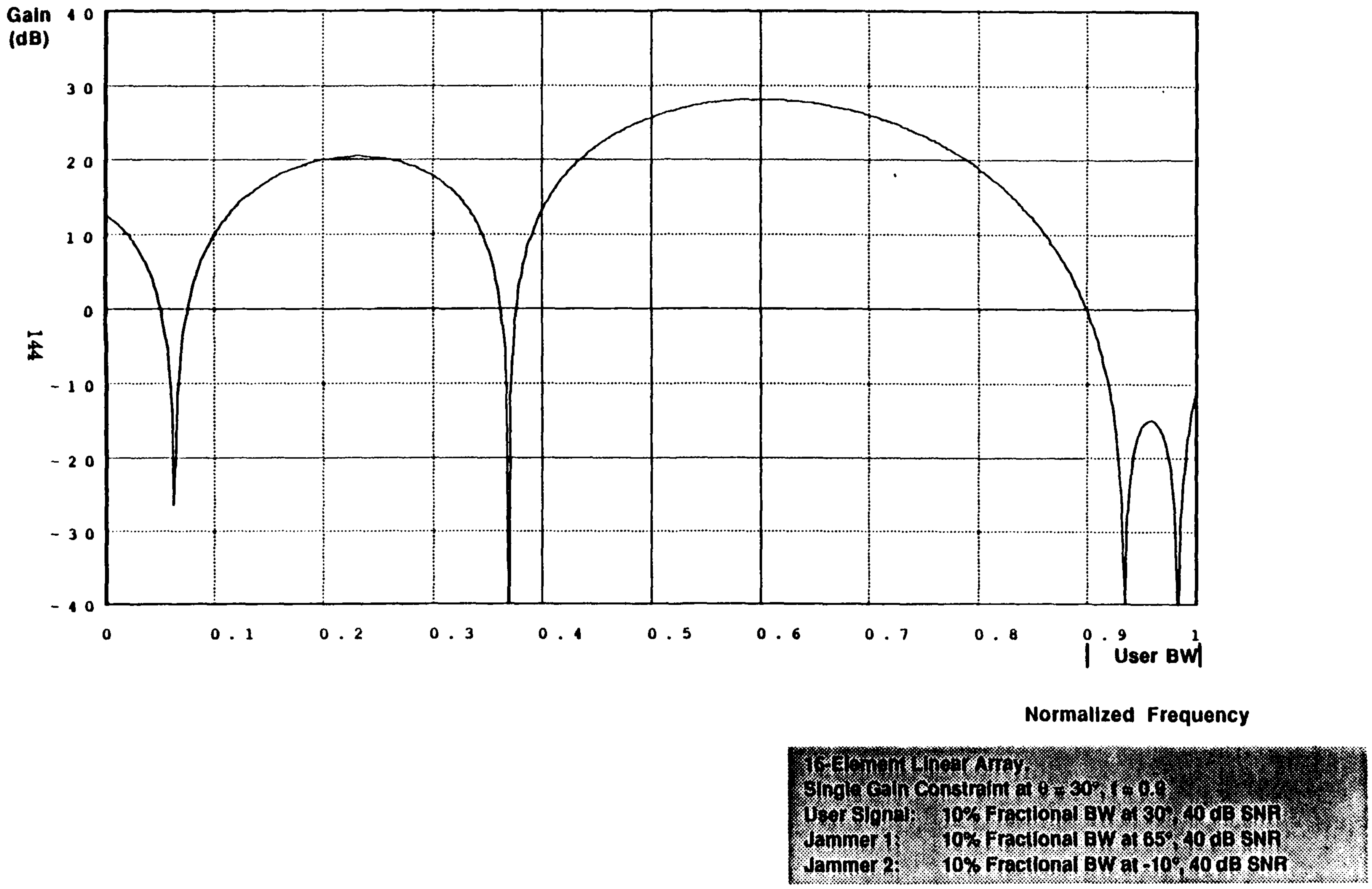


Figure 4.8: Frequency Response of a Linear Array in the Look Direction of the 10% Fractional BW User at 30 Degrees (Example 4.3)



One solution to the problem of using a narrowband array to pass a broadband user signal without distortion while rejecting broadband jammers is to impose *multiple constraints* to assure maintainance of gain across the user frequency band, rather than at a single frequency. This is equivalent to constraining gain across the angular sector from which the user signal apparently is arriving. Numerous techniques are available for constraint design, including response-sampling, derivative control, and eigenvector-based methods.

If Q linear constraints are employed, then the constraints equations can be collected into a single matrix equation of the form

$$C^H = \vec{f} \tag{4.13}$$

where C is a $K \times Q$ matrix and F is a $Q \times 1$ vector.

A variety of approaches is available for solving the associated constrained power-minimization problem. For example, the weight vector can be calculated directly from the generalized SMI expression

$$\vec{w} = R_{xx}^{-1} C [C^H R_{xx}^{-1} C]^{-1} \vec{f} \tag{4.14}$$

Alternatively, one can utilize the Generalized Sidelobe Canceller (GSC) processing structure which involves calculating 1) the minimum-norm solution

$$\vec{w}_o = C(C^H C)^{-1} \vec{f} \tag{4.15}$$

of the equation (4.13) and utilizing $\vec{w}_0$ as a quiescent weight vector, and then 2) suppressing the jammers by suitably adding vectors to $\vec{w}_0$ which are orthogonal to the columns of C which (therefore) do not disrupt satisfaction of the constraints.

The following example illustrates the use of multiple constraints to improve array performance for a broadband user.

Example 4.4: Performance of a Narrowband (Weight-and-Sum) Processor with Multiple Constraints in the Presence of Broadband Jammers and Broadband User [Linear Array]

Consider again the jamming scenario of the preceding example. However, assume that multiple constraints are imposed across the user signal frequency band. Six constraints, spaced evenly over the user frequency band of $f = 0.9 - 1.0$, were selected to provide unity gain and flat group delay. Group delay is defined as the slope of the phase response with respect to frequency. The units of group delay have been normalized such that the value 1.0 corresponds to 1 time sample (where the time sampling rate is assumed to correspond to twice the highest frequency present). The value of group delay delay was selected in this example as one sample.

The weight set that minimizes power subject to these six constraints produces the antenna pattern shown in Figure 4.9 at the frequency $f = 0.9$. Clearly, the weight-and-sum beamformer passes signals both in the actual and the apparent user directions while maintaining nulls on the two jammers. Figure 4.10 depicts the corresponding frequency response in the actual user direction. Clearly, the gain is flat and the group delay is approximately constant across the user frequency band. Therefore, the array now passes the user signal with very little distortion.

End of Example 4.4

Thus, provided that suitable multiple constraints are utilized to ensure flat gain and group delay across the user frequency band, constrained power-minimization leads to effective nulling weights for broadband scenarios.

The following example illustrates weight selection via constrained power minimization for a two-dimensional array.

Example 4.5: Performance of a Narrowband (Weight-and-Sum) Processor Array with Multiple Constraints in a Broadband Scenario [Planar Array]

Assume that a 6x6 uniform planar array of sensors is illuminated by broadband emitters with characteristics described in the following table:

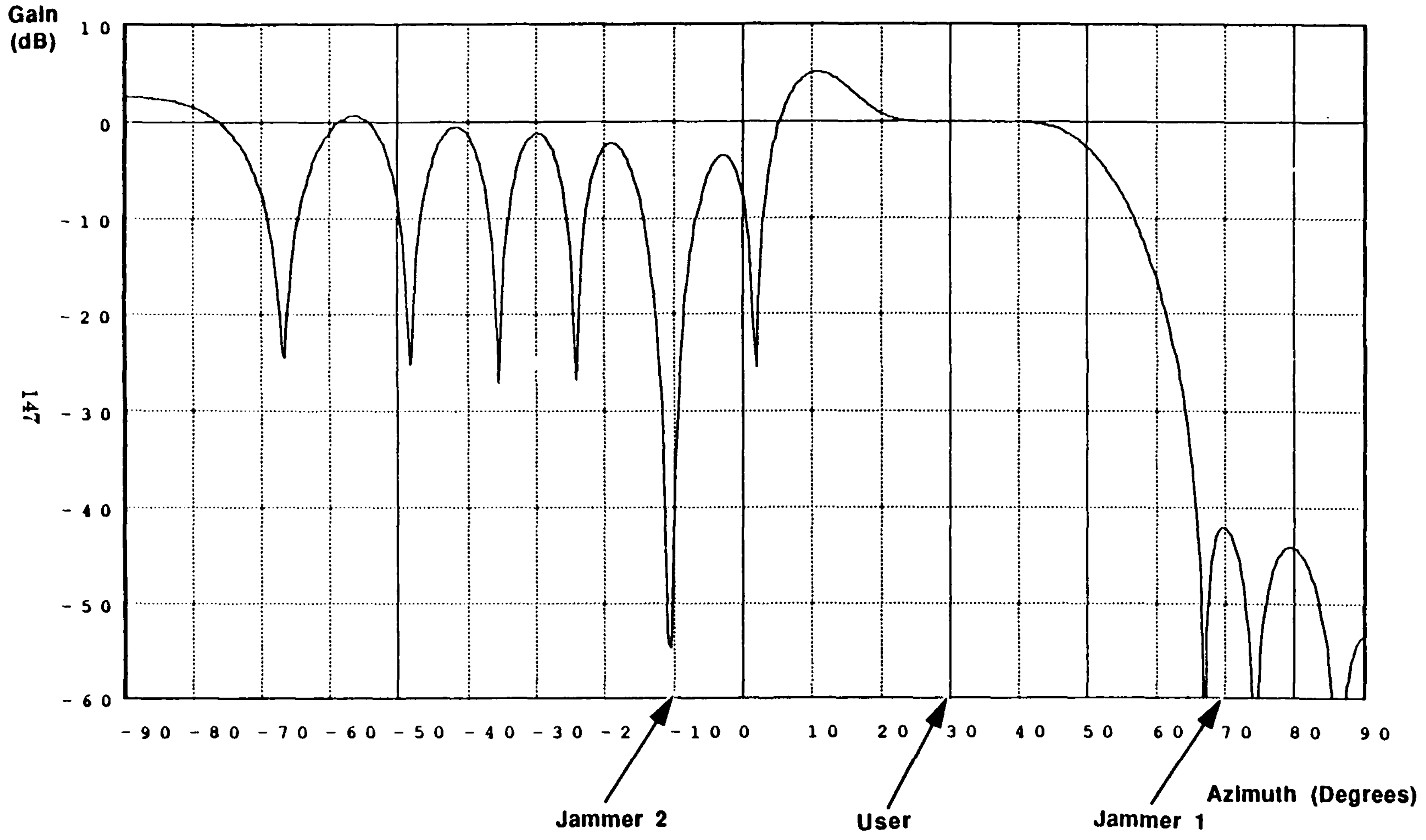
Signal	Azimuth	Elevation	Freq.	SNR, JNR (dB)
User	$+30°$	$60°$	$0.9 - 1.0$	40
Jammer 1	$+140°$	$60°$	$0.9 - 1.0$	40
Jammer 2	$-80°$	$60°$	$0.9 - 1.0$	40

Table 4.3: Parameters for the Two-Dimensional Broadband Scenario of Example 4.5

Constraints were selected as follows to assure constant gain and group delay across the user frequency band in the user direction. Six constraints, spaced evenly over the user frequency band of $f = 0.9 - 1.0$, were selected to provide unity gain and flat group delay for the user signal. The value of the group delay was chosen as one sample.

Figures 4.11 and 4.12 illustrate azimuth and elevation cuts of the antenna pattern for the

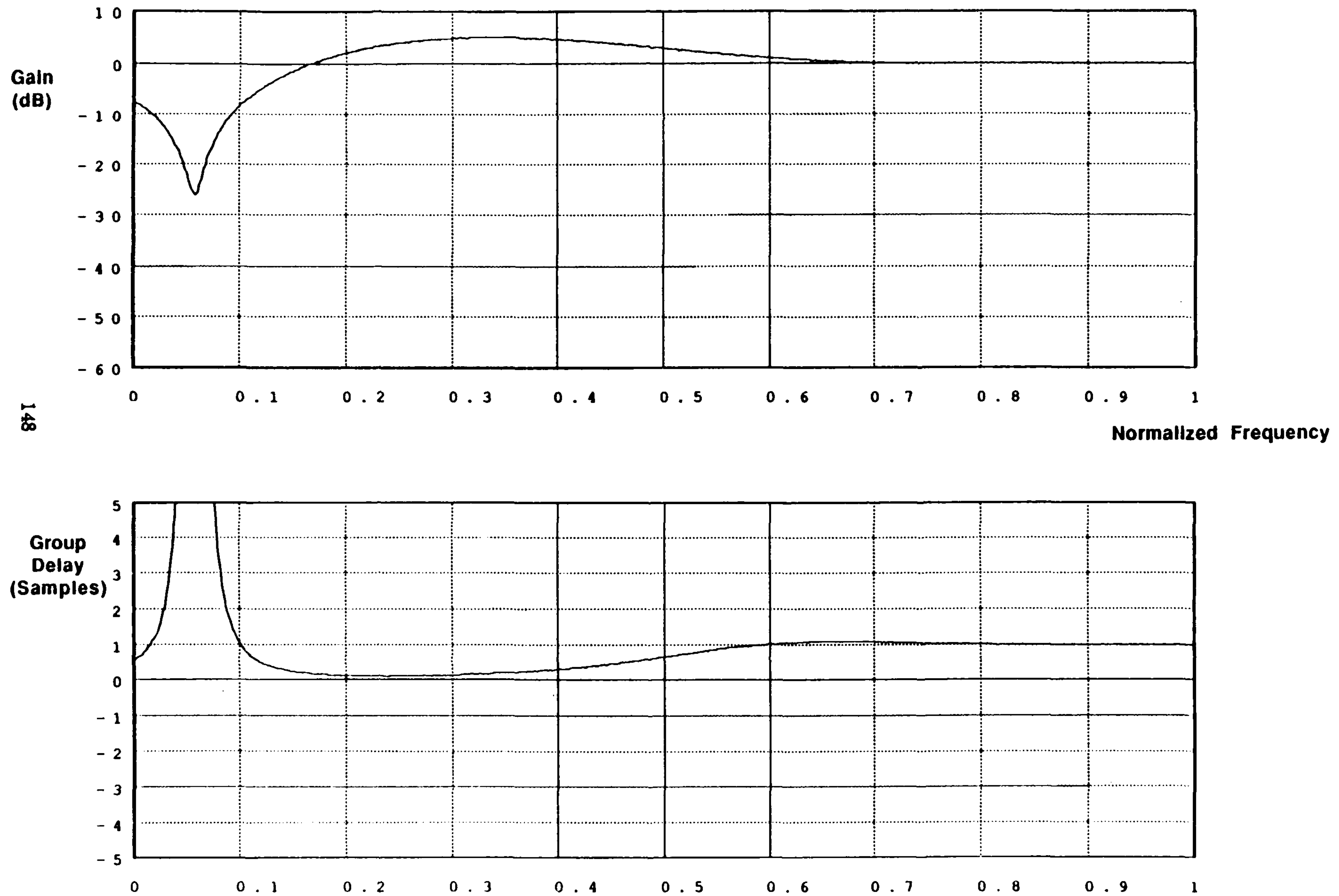
Figure 4.9: Beampattern (at f = 0.9) for a Linear Array After Adapting in the Presence of a Broadband User and Two Broadband Jammers Multiple Gain Constraints (Example 4.4)



16-Element Linear Array
6 Gain/Phase Constraints Across User Frequency Band.

User Signal:	10% Fractional BW at 30°; 40 dB JNR.
Jammer 1:	10% Fractional BW at 65°; 40 dB JNR.
Jammer 2:	10% Fractional BW at -10°; 40 dB JNR.

Figure 4.10: Frequency Response and Group Delay in the Look Direction of the 10% Fractional BW User at 30 Degrees. Multiple Constraints (Example 4.4)



SCENARIO: SEE FIGURE 4.9

constrained power-minimization weight set at the frequency $f = 0.9$. Clearly, the pattern passes signals not only from the actual user direction, but also from neighboring direction both in azimuth and elevation coordinates. Also, as one expects, the patterns exhibit discrete nulls on the jammers in the azimuth coordinate, but exhibit extended (sectoral) nulls on the jammers in the elevation coordinate.

Figure 4.13 depicts the frequency response (magnitude and group delay) of the array in the actual user direction. It is evident from the figure that the constraints have been effective in providing both constant gain and group delay across the user frequency band.

End of Example 4.5

4.2.3 Multiple Nulls

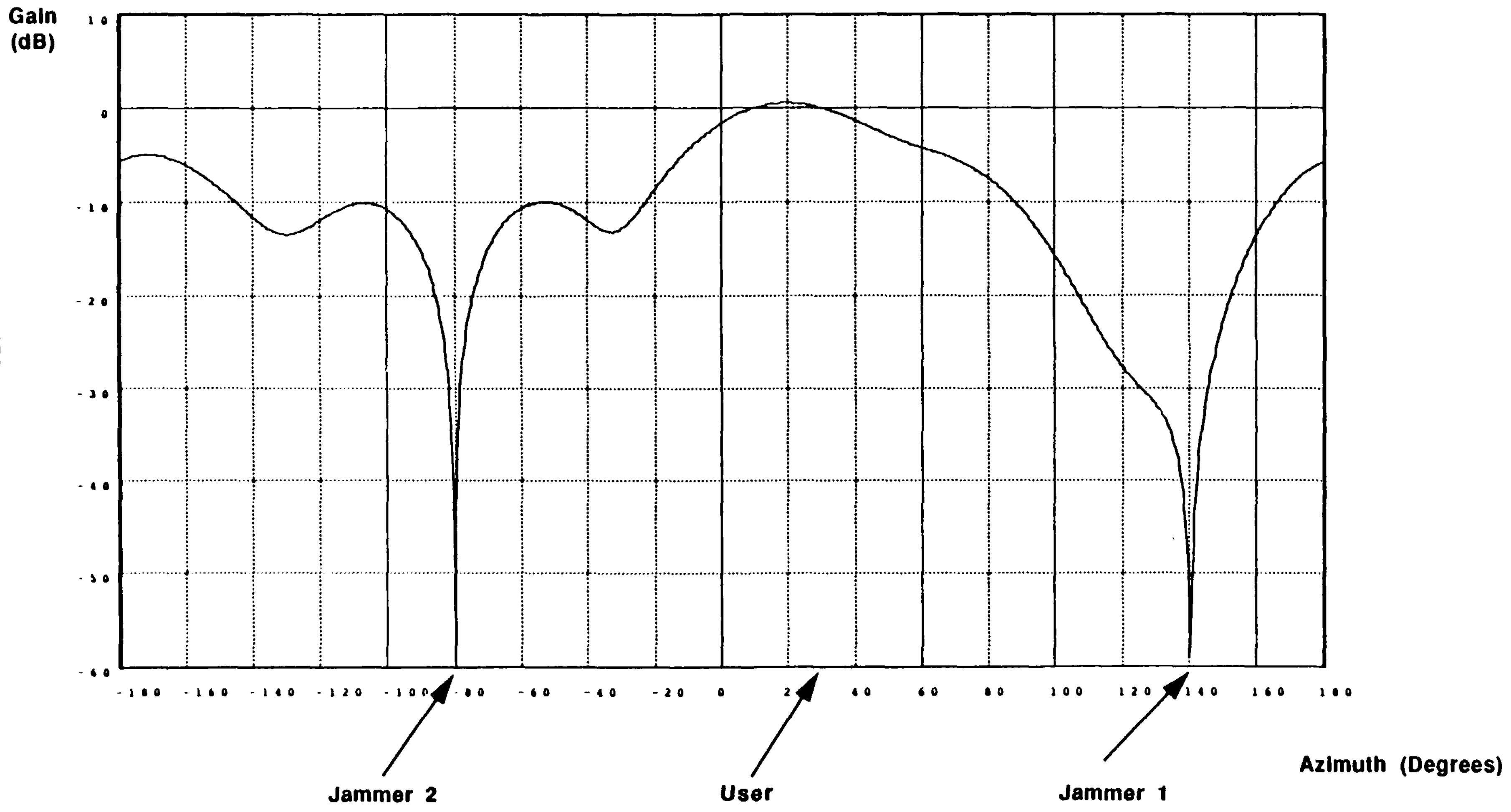
Techniques which solve the Wiener-Hopf equation, or related equations for a narrowband array, typically place a single null on each narrowband jammer and place a sequence of nulls on a broadband jammer thereby providing suppression over an angular sector. We have developed techniques which automatically place multiple nulls on (or near) narrowband and broadband jammers. These methods will be referred to as *multiple null techniques*[10].

Three specializations of the multiple null concept were studied. First, multiple nulls can be placed directly on top of each other thereby forcing not only the gain in the jammer direction to be zero but also the higher order derivatives of the pattern to be zero. This technique is referred to as ***higher-order nulls***. Second, multiple nulls can be placed in the vicinity of each jammer. This technique is referred to as *clustered nulls*. Finally, in the case of a planar array, nulls can be automatically placed along a line in elevation, also corresponding to different frequency components of the jammer arriving from angular position. This technique is referred to as ***elevation*** or radial nulls.

From a spectral point of view, multiple nulls extend the frequency band in which jammer signals are suppressed beyond the band actually occupied by the jamming signals. This feature potentially is useful in applications in which one wishes to avoid the nulling system problems associated with a user signal. (See Chapters 3 and 5.) Specifically, the opportunity exists to filter out the user signal by means of bandpass filters, and then develop nulling weights based upon the jammer signal present in the adjacent bands. The nulling weights then also suppress the inband components of the jamming signal.

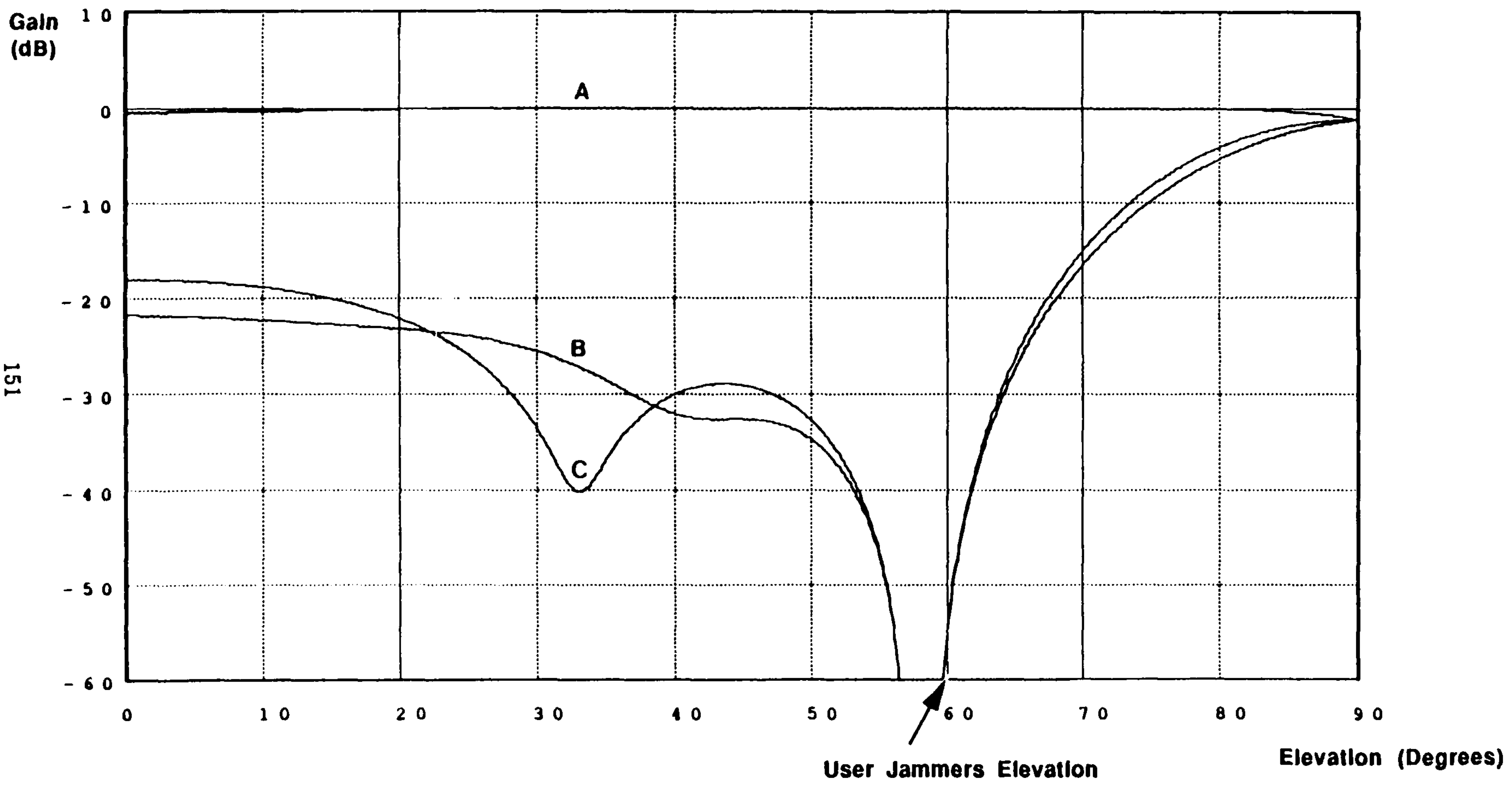
From a spatial point of view, multiple nulls extend the angular sector both in azimuth

Figure 4.11: Beampattern (at f = 1.0) for a Planar Array With Multiple Constraints
Elevation = 60 Degrees (Example 4.5)



6 x 6 Planar Array
6 Gain Constraints Across User Band
User Signal: 10% Fractional BW at AZ/EL of 30°/60°. JNR = 40 dB
Jammer 1: 10% Fractional BW at AZ/EL of 140°/60°. JNR = 40 dB
Jammer 2: 10% Fractional BW at AZ/EL of -80°/60°. JNR = 40 dB

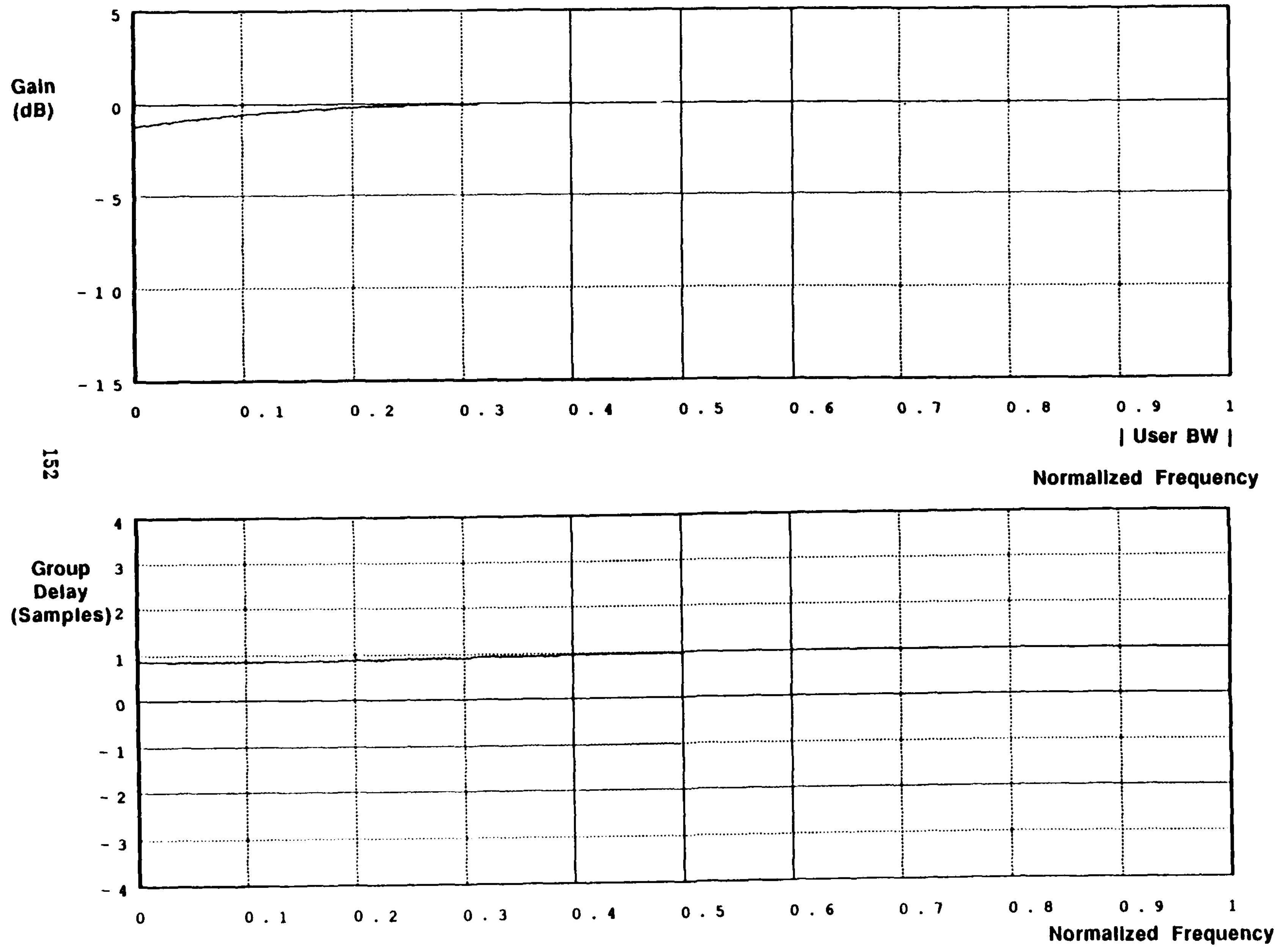
Figure 4.12: Beampattern (at f = 1.0) for a Planar Array With Multiple Constraints Azimuth Cuts at 30°, 140°, -80°



CURVES:
A. User Azimuth Cut
B. Jammer 1 Azimuth Cut
C. Jammer 2 Azimuth Cut

6 x 6 Planar Array
6 Gain Constraints Across User Band
User Signal: 10% Fractional BW at AZ/EL of 30°/60°. JNR = 40 dB
Jammer 1: 10% Fractional BW at AZ/EL of 140°/60°. JNR = 40 dB
Jammer 2: 10% Fractional BW at AZ/EL of -80°/60°. JNR = 40 dB

Figure 4.13: Frequency Response and Group Delay (In User Look Direction) for a Planar Array With Multiple Constraints (Example 4.5)



and elevation over which jammer suppression is provided. This feature potentially is useful when the nulling system is mounted on a platform which exhibits small directional oscillations.

4.2.4 Producing Higher-Order Nulls (Linear Arrays)

Consider a linear array of omni-directional sensors with arbitrary spacing. For a given weight vector $\vec{w}$, we define the gain $G(\theta)$ in the direction θ to be

$$G(\theta) = \vec{w}^H \vec{u}(\theta) \tag{4.16}$$

where $\vec{u}(\theta)$ denotes the $K \times 1$ unit-arrival vector for a plane wave incident upon the array from an off-broadside angle θ; that is

$$\vec{u}(\theta) = [e^{jx_1\alpha}, e^{jx_2\alpha}, \cdots e^{jx_K\alpha}]^T/\sqrt{N} \tag{4.17}$$

where x_k denotes the distance to the k^{th} sensor measured from an arbitrary phase- reference point on the array axis, and

$$\alpha = (2\pi/\lambda)\sin\theta \tag{4.18}$$

with the wavelength λ.

We define the pattern $G(\theta)$ to have an M^{th} higher-order null in the direction θ_0 provided the following conditions are satisfied

$$G(\theta_0) = 0 \tag{4.19}$$

and

$$G(\theta_0 + \Delta\theta) = \alpha_M(\Delta\theta)^M \tag{4.20}$$

with $\Delta_M \neq 0$, for small perturbations $\Delta\theta$ from the direction θ_0. Note that condition (4.19) requires that the gain be zero in the direction θ_0. Condition (4.20) requires that the gain also be near-zero for neighboring directions.

The conditions (4.19) and (4.20) are equivalent to the following conditions on $G(\theta)$ and its derivatives

$$\{\frac{d^m}{d\theta^m}G(\theta)\}|_{\theta=\theta_0} = 0 \tag{4.21}$$

for $0 \leq m \leq M-1$. That is, both the gain and $M-1$ higher derivatives are zero in the direction θ_0. Use of (4.21) in (4.16) shows that $\vec{w}$ produces a M^{th} order null in the direction θ_0 provided the weight vector $\vec{w}$ satisfies

$$\vec{w}^H \vec{u}(\theta_0) = 0 \tag{4.22}$$

and

$$\vec{w}^H\{\frac{d^m}{d\theta^m}G(\theta)\}_{\theta=\theta_0} = 0 \tag{4.23}$$

for $1 \leq m \leq M-1$. It is evident from (4.22) and (4.23) that $\vec{w}$ must expend M of its available $K-1$ degrees of freedom for each M^{th} order null that it produces.

With some algebra, it can be shown that the conditions (4.22) and (4.23) are equivalent to the single condition

$$\vec{w}^H[A^m\vec{u}(\theta_0)] = 0 \tag{4.24}$$

for $m = 0, 1, \cdots M-1$, where

$$A = \begin{pmatrix} x_1 & & & 0 \\ & x_2 & & \\ & & \cdot & \\ & & & \cdot \\ 0 & & & x_K \end{pmatrix} \tag{4.25}$$

and x_k is the coordinate of the k^{th} sensor of the array relative to an arbitrary reference point along the array axis.

Thus, if the weight vector $\vec{w}$ is to produce an M^{th} order null on all J interference sources, then $\vec{w}$ must satisfy the conditions

$$\vec{w}^H[A^m\vec{u}_i] = 0 \tag{4.26}$$

for $m = 0, 1, \cdots M-1$ and $i = 1, 2, \cdots J$ where $\vec{u}_i$ denotes the unit-arrival vector for the i^{th} jammer. Equation (4.26) provides the key to producing higher-order nulls on jammers. Note that (4.26) constitutes a set of *orthogonality conditions* for the weight vector $\vec{w}$.

A representative snapshot for a uniform linear array is of the form

$$\vec{x}(n) = \sum_{i=1}^{J} a_i(n)\vec{u}_i + \vec{\epsilon}(n) \tag{4.27}$$

The objective of conventional nulling algorithms is to construct a weight vector $\vec{w}$ which is orthogonal to the $\vec{u}_i$ for the jammers.

In high JNR situations, the weight set can be made orthogonal to the set of vectors $\{A^m\vec{u}_i\}$ by providing the adaptive algorithm with independent snapshots, $\{A^m\vec{x}(n)\}$, for $m = 0, 1, \cdots M-1$. Thus, higher-order nulls can be achieved by producing a series of fictitious snapshots $A\vec{x}(n), A^2\vec{x}(n), \cdots A^{M-1}\vec{x}(n)$ for each actual snapshot $\vec{x}(n)$ received.

More generally, if the sample covariance matrix is constructed as follows:

$$\hat{R}_{xx} = \frac{1}{K}\sum_{n=0}^{K-1} \vec{x}(n)\vec{x}^H(n) \tag{4.28}$$

then providing the adaptive algorithm with a new sample covariance $\tilde{R}_{xx}$ matrix defined by

$$\tilde{R}_{xx} = \frac{1}{K}\sum_{n=0}^{K-1} \vec{x}(n)\vec{x}^H(n) + \frac{1}{K}\sum_{n=0}^{K-1} A^1\vec{x}(n)\vec{x}^H(n)(A^1)^H + \cdots \frac{1}{K}\sum_{n=0}^{K-1} A^{M-1}\vec{x}(n)\vec{x}^H(n)(A^{M-1})^H \tag{4.29}$$

forces an M^{th} zero on each jammer. This covariance matrix can also be written as

$$\tilde{R}_{xx} = \sum_{m=0}^{M-1} A^m \hat{R}_{xx}(A^m)^H \tag{4.30}$$

Thus, for SMI and related algorithms, higher-order nulls can be achieved by replacing the actual sample covariance matrix by the modified sample matrix $\tilde{R}_{xx}$ given in equation (4.30).

Minimization of power subject to a gain constraint in the user direction gives the classical Wiener-Hopf equation for the weight vector. Inclusion of M^{th} higher-order nulls within this framework yields the following equation for the weight vector

$$\vec{w} = \mu[\sum_{m=0}^{M-1} A^m R_{xx}(A^m)^H]^{-1}\vec{u}_0 = \mu\tilde{R}_{xx}^{-1}\vec{u}_0 \tag{4.31}$$

Multiple gain/phase constraints along with higher-order nulls can be achieved by utilizing the following equation for the weight set

$$\vec{w} = \tilde{R}_{xx}^{-1}C[C^H\tilde{R}_{xx}^{-1}C]^{-1}\vec{f} \tag{4.32}$$

The following example illustrates the use of high-order nulls in linear arrays.

Example 4.6: Use of Higher-Order Nulls in Linear Arrays

Consider a scenario consisting of a 16-element array with equally spaced omni-directional elements. We assume that two uncorrelated narrowband jammers illuminate the array from angles of $-50°$ and $20°$ and that the JNR for each jammer is 40 dB. The user signal is assumed to be absent and the array is steered in a $5°$ look direction. The steady-state Wiener equation was solved for the adaptive weights $\vec{w}$ and the zeros of the associated weight polynomial

$$w(z) = \sum_{k=1}^{K} w_k z^k \tag{4.33}$$

were computed. In this equation, z is defined

$$z = e^{j\pi \sin\theta} \tag{4.34}$$

where θ represents physical arrival angle with respect to array boresight.

Figure 4.14a depicts the locations of the zeros in the complex z-plane. Note that there are 15 zeros present (corresponding to a length 16 weight polynomial) and that the adaptive processor has placed a single zero in direction of each jammer.

Figure 4.14b depicts the zero locations after higher-order null processing. Higher-order nulls of degree three were achieved by using a modified covariance matrix $\tilde{R}_{xx}$ defined by

$$\tilde{R}_{xx} = R_{xx} + \alpha A R_{xx} A^H + \alpha^2 A^2 R_{xx} (A^2)^H \tag{4.35}$$

where A is a diagonal matrix with a ramp along the diagonal. The constant α was chosen so as to make the ramp have unity norm. Note that the processing does not require explicit jammer directional information. The modified covariance matrix was used in the Wiener equation to compute the weights and the zeros of the associated polynomial were determined. As can be seen in Figure 4.14b, two additional zeros have been placed on each jammer and the remaining $15 - 6 = 9$ zeros remain spaced about the unit circle.

Figure 4.15 depicts the beampattern for the weight set, and the first and second beampattern derivatives with respect to $\sin\theta$. As can be seen, the weight set produces an antenna pattern which has zero gain, zero first derivative, and zero second derivative in the directions of the jammers.

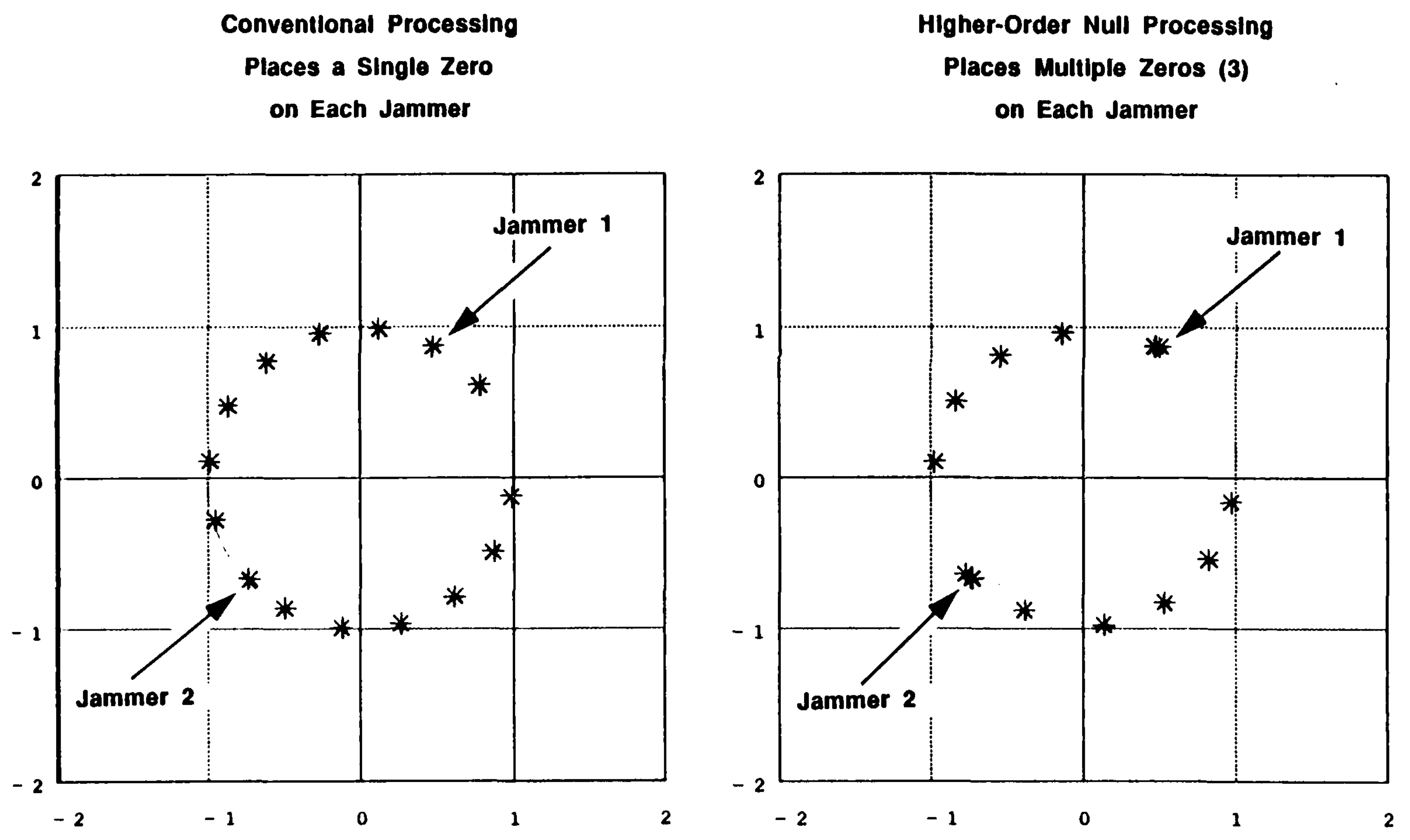
End of Example 4.6

4.2.5 Producing Clustered Nulls (Linear Arrays)

Clustered nulls can be achieved by replacing the matrix A^i with a diagonal matrix $T^{(i)}$, which has an nth diagonal element given by $e^{jn\pi\delta_i}$. In this case each snapshot is *modulated* so that it appears to the adaptive processor as having arrived from a different (nearby) angle.

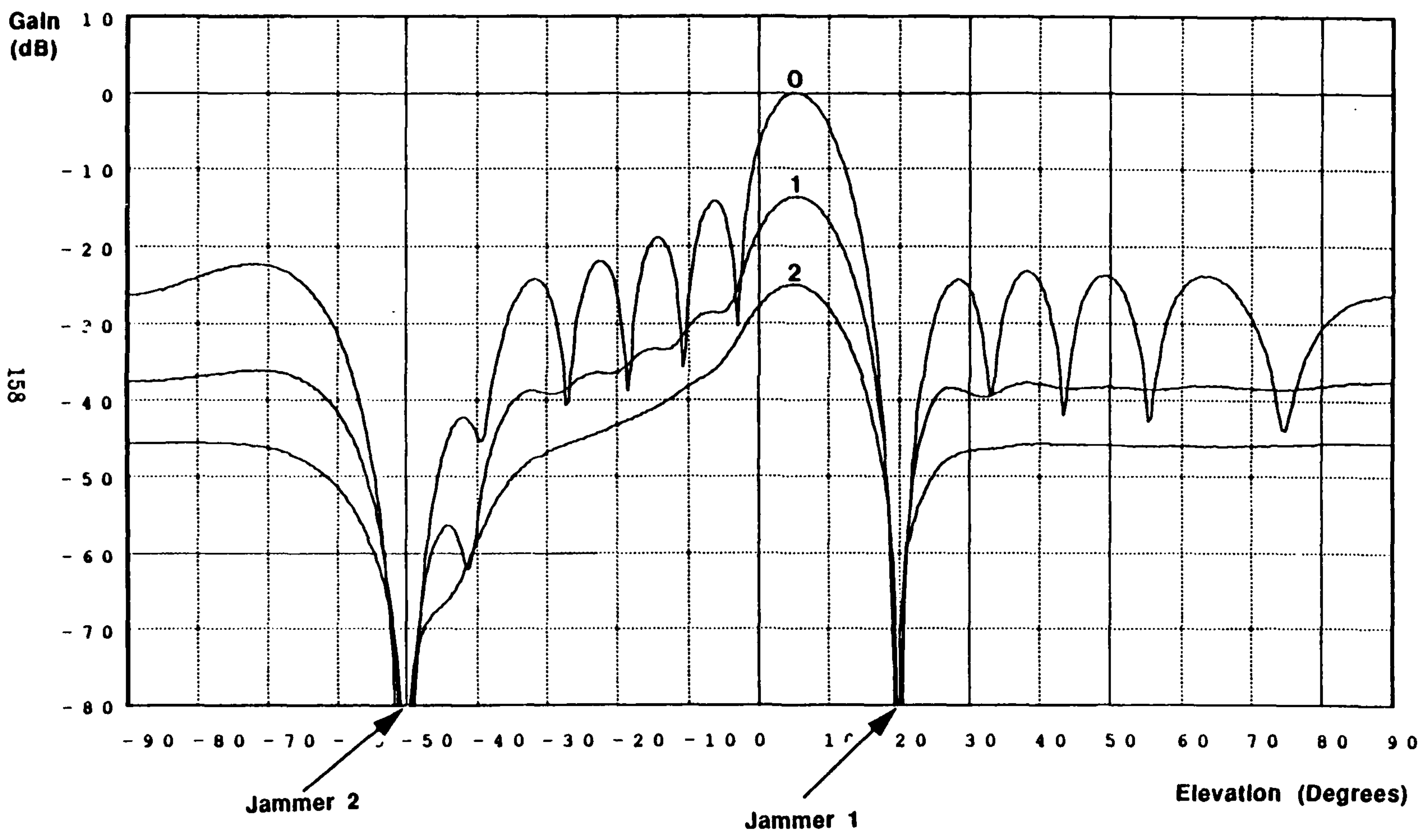
Multiple $T^{(i)}$ matrices can be used so as to synthesize a series of snapshots corresponding to arrivals from synthetic emitters *near* the actual emitters. In this case the adaptive processor will sense (in the high JNR case) arrivals coming from both the actual set of jammer directions $\{\theta_j\}$ as well as from the set of modified (translated) directions $\{\sin^{-1}(\delta_i +$

Figure 4.14: Locations of Zeros in Complex Z-Plane Without and With Higher Order Null Processing (Example 4.6)



16 Element Linear Array Steered Toward 5°
40 dB JNR Jammers at 20°, -50°

Figure 4.15: Beampattern and First Two Derivatives for a Linear Array With Higher-Order Null Processing (Example 4.6)



CURVES:
0. Beampattern
1. First Derivative of Beam Pattern
2. Second Derivative of Beam Pattern

16 Element Linear Array
40 dB JNR Jammers at 20°, 50°

$\sin\theta_j)\}$. Note that the translation in physical angle is a function of the angle itself - arrivals from larger absolute angles will be translated by larger amounts.

Similarly, by providing the adaptive algorithm with a new sample covariance matrix given by

$$\tilde{R}_{xx} = \sum_{i=0}^{M-1} T^{(i)} \hat{R}_{xx} (T^{(i)})^H \tag{4.36}$$

it is possible to achieve M clustered nulls near each jammer, provided that sufficient degrees of freedom are available.

Clustered nulls, with each null translated by an electrical angle of $\pi\delta_i$, can be achieved by computing the weight set

$$\vec{w} = \mu[\sum_{i=0}^{M-1} (T^{(i)})^H R_{xx} T^{(i)}]^{-1} \vec{u}_0 = \mu \tilde{R}_{xx}^{-1} \vec{u}_0 \tag{4.37}$$

Multiple gain/phase constraints along with clustered nulls can be achieved by utilizing the following equation for the weight set

$$\vec{w} = \tilde{R}_{xx}^{-1} C [C^H \tilde{R}_{xx}^{-1} C]^{-1} \vec{f} \tag{4.38}$$

The following example illustrates the use of clustered nulls for linear arrays.

Example 4.7: Use of Clustered Nulls for Linear Arrays

Consider the same scenario as in Example 4.6. Figure 4.16a depicts the locations of the zeros in the complex-z plane for the Wiener weight set. Figure 4.16b depicts the zero locations after clustered-null processing.

Three clustered nulls were achieved by utilizing a covariance matrix $\tilde{R}_{xx}$ equal to

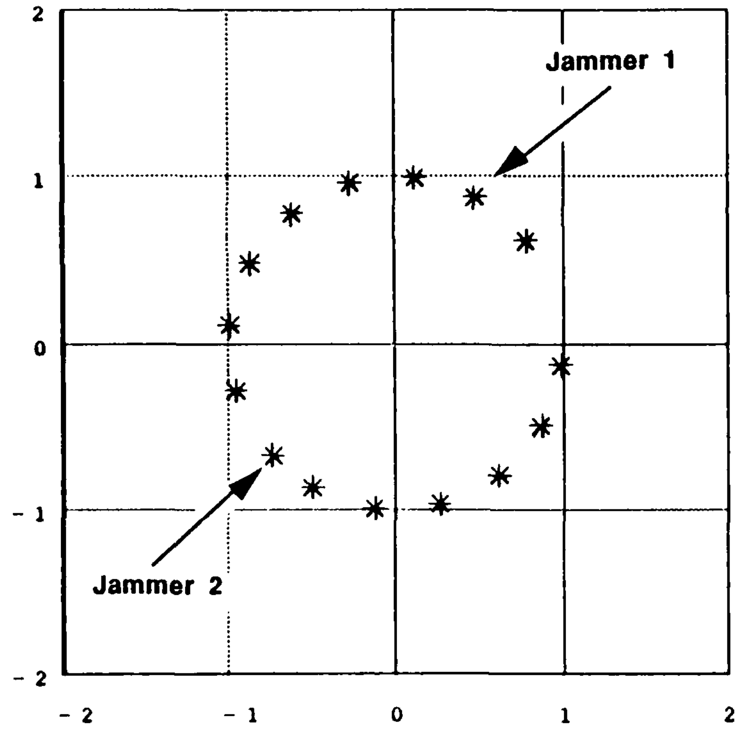
$$\tilde{R}_{xx} = R_{xx} + T^{(1)} R_{xx} (T^{(1)})^H + T^{(2)} R_{xx} (T^{(2)})^H \tag{4.39}$$

where $T^{(1)}$ was a diagonal matrix with the n^{th} diagonal element given by $e^{jn\pi\delta_1}$. The factor δ_1 was selected as $\delta_1 = \sin 2^\circ = +0.0349$ corresponding to a $+2^\circ$ shift at broadside. The matrix $T^{(2)}$ was a diagonal matrix with the n^{th} diagonal element given by $e^{jn\pi\delta_2}$. The factor δ_2 was selected as $\delta_2 = \sin(-2^\circ) = -0.0349$ corresponding to a -2° shift at broadside. Note that the processing does not require explicit jammer directional information.

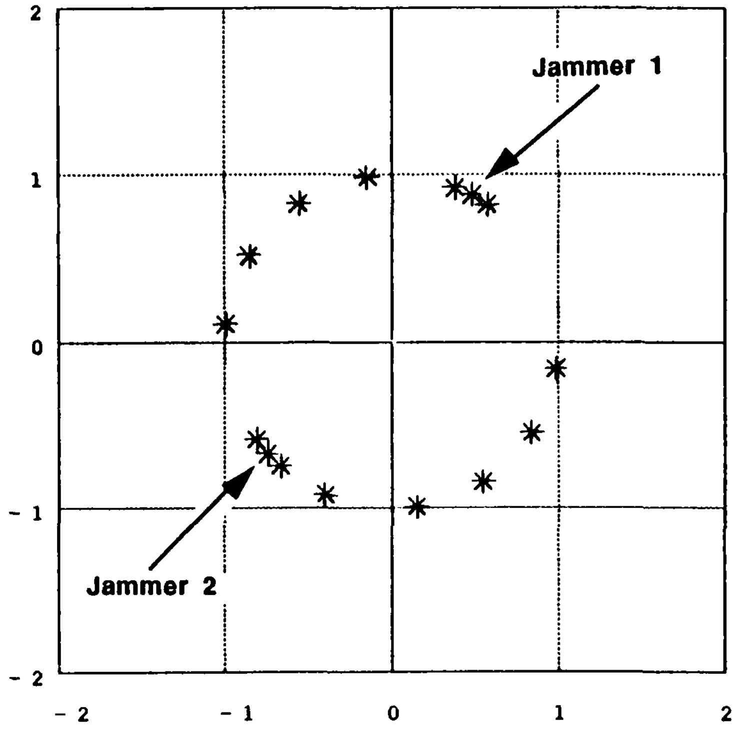
The modified covariance matrix (4.39) was used in the Wiener equation to compute the weights, using equation (4.37), and the zeros of the associated polynomial were determined. As can be seen in Figure 4.16b, three zeros are clustered around each of the jammers present.

Figure 4.16: Locations of Zeros in Complex Z-Plane
Without and With Clustered Null Procesing (Example 4.7)

Conventional Processing
Places a Single Zero
on Each Jammer



Clustered Null Processing
Places Clustered Zeros (3)
on Each Jammer



16 Element Linear Array
40 dB JNR Jammers at 20°, -50°

One zero is placed directly on each jammer, one zero is placed at a slightly larger arrival angle, and one zero is placed at a slightly smaller arrival angle. The spread of the clustered zeros in the z plane is identical for both jammers. In physical angle however, the spread of the clustered zeros is larger for the jammer located at the larger arrival angle.

Figure 4.17 shows the beampattern corresponding to the clustered null processing. The clustered nulls at 20° and −50° are clearly visible. Note that the nulls are spaced closer together for the 20° jammer as contrasted with the 50° jammer. The spread between the clustered nulls near the 20° jammer is given by

$$|\sin^{-1}[\delta_1 + \sin(20^\circ)] - \sin^{-1}[\delta_2 + \sin(20^\circ)]| = 4.26^\circ \tag{4.40}$$

while the spread between the clustered nulls near the −50° jammer is given by

$$|\sin^{-1}[\delta_1 + \sin(-50)^\circ] - \sin^{-1}[\delta_2 + \sin(-50)^\circ]| = 6.24^\circ \tag{4.41}$$

End of Example 4.7

4.2.6 Producing Higher-Order Nulls (Planar Arrays)

Consider next a planar array of K omni-directional sensors. For a given weight vector $\vec{w}$, we define the gain $G(\theta,\phi)$ of the array in the direction specified by the azimuth, elevation coordinates (θ,ϕ) to be

$$G(\theta,\phi) = \vec{w}^H \vec{u}(\theta,\phi) \tag{4.42}$$

where the $K \times 1$ vector $\vec{u}(\theta,\phi)$ is a unit-arrival vector corresponding to a planar wave incident from the direction (θ,ϕ). Note that K represents the total number of elements in the planar array.

We will say that the array produces an M^{th} order null in the direction specified by the coordinates θ_0 and ϕ_0 provided the gain function satisfies the following conditions

$$G(\theta_0,\phi_0) = 0 \tag{4.43}$$

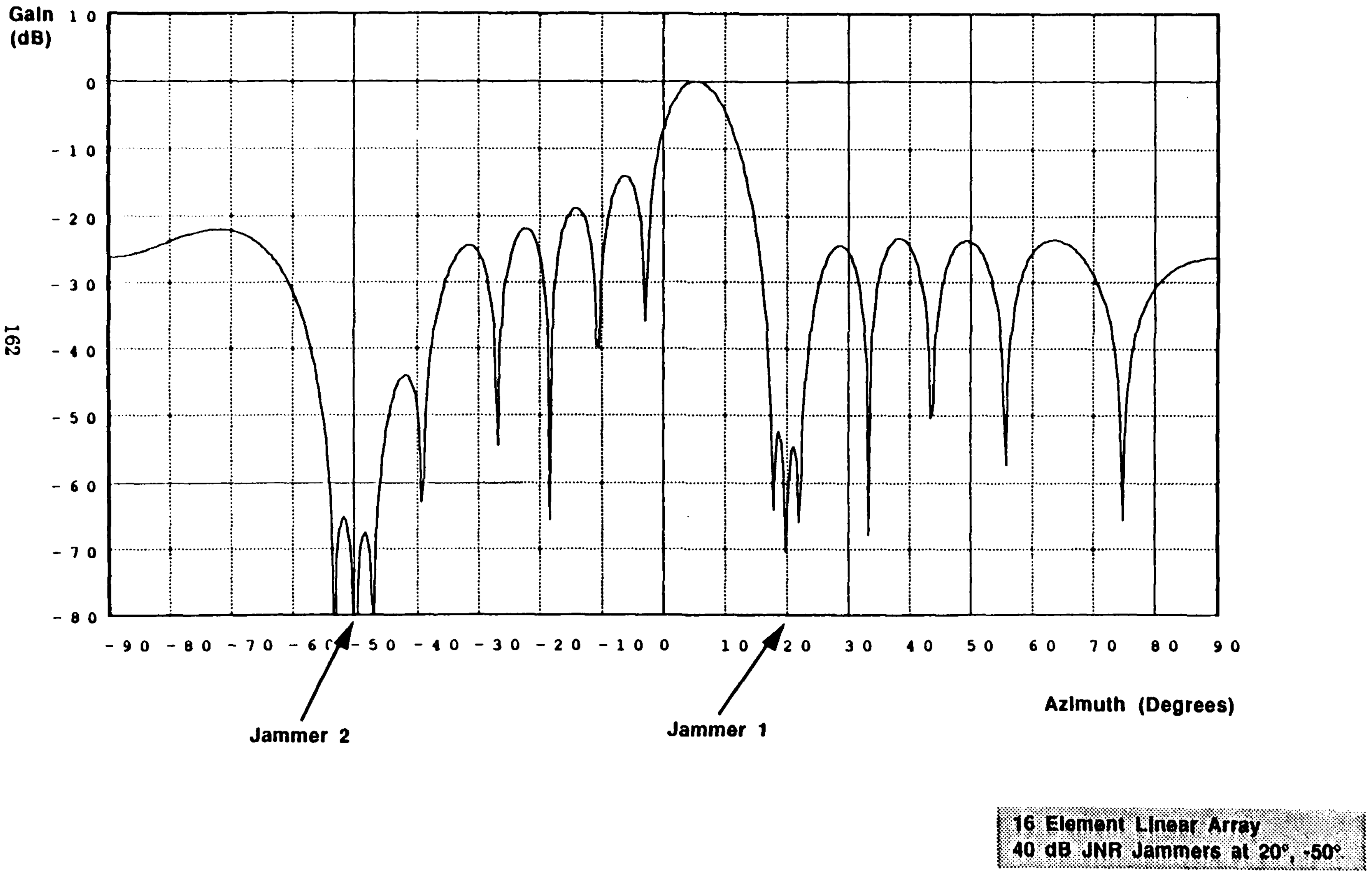
and

$$G(\theta_0 + \Delta\theta, \phi_0 + \Delta\phi) = \sum_{m=0}^{M} \alpha_m (\Delta\theta)^m (\Delta\phi)^{M-m} \tag{4.44}$$

for small perturbation $\Delta\theta$ and $\Delta\phi$ from θ_0 and ϕ_0. Conditions (4.43) and (4.44) require that the gain partial derivatives equal zero as follows

$$\frac{\partial^{M'} G(\theta,\phi)}{\partial\theta^m \partial\phi^{M'-m}}\Big|_{\theta=\theta_0,\phi=\phi_0} = 0 \tag{4.45}$$

Figure 4.17: Beampattern for a Linear Array
With Clustered Null Processing (Example 4.7)



for $m = 0, 1, \cdots M'$, and for each integer M' such that $1 \leq M' \leq (M-1)$.

Use of (4.42) in (4.45) shows that the weight set $\vec{w}$ produces an M^{th} order null in the direction (θ_0, ϕ_0) provided it satisfies the conditions

$$\vec{w}^H \vec{u}(\theta, \phi) = 0 \tag{4.46}$$

and

$$\vec{w}^H \{\frac{\partial^{M'} \vec{u}(\theta, \phi)}{\partial\theta^m \partial\phi^{M'-m}}\}|_{\theta=\theta_0, \phi=\phi_0} = 0 \tag{4.47}$$

for $m = 0, 1, \cdots M'$, and for each integer M' such that $1 \leq M' \leq (M-1)$. It is evident from (4.46) and (4.47) that $\vec{w}$ must expend

$$1 + \sum_{M'=1}^{M-1} (M'+1) = M(M+1)/2 \tag{4.48}$$

degrees of freedom in order to produce each M^{th} degree null.

The conditions (4.46) and (4.47) can be made more explicit by expressing unit arrival vectors in Cartesian coordinates. Specifically, the component of the unit arrival vector from azimuth angle θ and elevation angle ϕ at sensor position x_m, y_n can be written as

$$u_{m,n}(\theta, \phi) = e^{j(\Omega_x x_m + \Omega_y y_n)} \tag{4.49}$$

where

$$\Omega_x = (2\pi/\lambda) \cos\phi \sin\theta \tag{4.50}$$

and

$$\Omega_y = (2\pi/\lambda) \cos\phi \cos\theta \tag{4.51}$$

The partial derivative of this component of $\vec{u}(\theta, \phi)$ with respect to Ω_x is

$$\frac{\partial u_{m,n}(\theta, \phi)}{\partial \Omega_x} = j x_m e^{j(\Omega_x x_m + \Omega_y x_n)} \tag{4.52}$$

and the partial derivative of this component of $\vec{u}(\theta, \phi)$ with respect to Ω_y is

$$\frac{\partial u_{m,n}(\theta, \phi)}{\partial \Omega_y} = j y_n e^{j(\Omega_x x_m + \Omega_y x_n)} \tag{4.53}$$

Thus, the partial derivative of this component with respect to θ is given by

$$\frac{\partial u_{m,n}(\theta, \phi)}{\partial\theta} = \frac{\partial u_{m,n}(\theta, \phi)}{\partial \Omega_x} \frac{\partial \Omega_x}{\partial \theta} + \frac{\partial u_{m,n}(\theta, \phi)}{\partial \Omega_y} \frac{\partial \Omega_y}{\partial \theta} \tag{4.54}$$

$$= \{j[(2\pi/\lambda) \cos\phi \cos\theta] x_m - j[(2\pi/\lambda) \cos\phi \sin\theta] y_n\} u_{m,n}(\theta, \phi) \tag{4.55}$$

Similarly, the partial derivative of this component with respect to ϕ is given by

$$\frac{\partial u_{m,n}(\theta,\phi)}{\partial \phi} = \frac{\partial u_{m,n}(\theta,\phi)}{\partial \Omega_x}\frac{\partial \Omega_x}{\partial \phi} + \frac{\partial u_{m,n}(\theta,\phi)}{\partial \Omega_y}\frac{\partial \Omega_y}{\partial \phi} \tag{4.56}$$

$$= \{-j[(2\pi/\lambda)\sin\phi\sin\theta]x_m - j[-(2\pi/\lambda)\sin\phi\cos\theta]y_n\}u_{m,n}(\theta,\phi) \tag{4.57}$$

Thus, we can express the partial derivatives of the unit arrival vector as

$$\frac{\partial \vec{u}(\theta,\phi)}{\partial \theta} = j\frac{\partial \Omega_x}{\partial \theta}A_x\vec{u}(\theta,\phi) + j\frac{\partial \Omega_y}{\partial \theta}A_y\vec{u}(\theta,\phi) \tag{4.58}$$

and

$$\frac{\partial \vec{u}(\theta,\phi)}{\partial \phi} = j\frac{\partial \Omega_x}{\partial \phi}A_x\vec{u}(\theta,\phi) + j\frac{\partial \Omega_y}{\partial \phi}A_y\vec{u}(\theta,\phi) \tag{4.59}$$

where A_x and A_y are diagonal matrices. Each element of the diagonal matrix A_x multiplies the corresponding component of the arrival vector $\vec{u}(\theta,\phi)$ by its offset along the x axis. Similarly, each element of the diagonal matrix A_y multiplies the corresponding component of the arrival vector $\vec{u}(\theta,\phi)$ by the its offset along the y axis.

For $M' = 1$ the condition

$$\vec{w}^H\frac{\partial \vec{u}(\theta,\phi)}{\partial \theta} = 0 \tag{4.60}$$

therefore takes the form

$$\vec{w}^H A_x\vec{u}(\theta,\phi) = 0 \tag{4.61}$$

and

$$\vec{w}^H A_y\vec{u}(\theta,\phi) = 0 \tag{4.62}$$

Extension of the foregoing reasoning shows that conditions (4.43) and (4.44) are equivalent to the single condition

$$\vec{w}^H(A_x)^m(A_y)^{M'-m}\vec{u}(\theta,\phi) = 0 \tag{4.63}$$

for $0 \le M' \le M-1$ and $m = 0,1\cdot\cdot M'$.

Thus, if $\vec{w}$ is to produce an M^{th} order null on all J interference sources, then $\vec{w}$ must satisfy the conditions

$$\vec{w}^H(A_x)^m(A_y)^{M'-m}\vec{u}_j(\theta,\phi) = 0 \tag{4.64}$$

for for $0 \le M' \le M-1$ and $m = 0,1\cdot\cdot M'$ and $j = 1,2\cdot\cdot J$

In high JNR situations, the weight set can be made orthogonal to the set of vectors $A_x^m A_y^{M'-m}\vec{u}_j$ by providing the adaptive algorithm with a series of snapshots, $A_x^m A_y^{M'-m}\vec{x}(n)$. With analogy to the linear array, higher-order nulls can be achieved by corresponding processing of the sample covariance matrix.

The following example illustrates the use of higher-order nulls in planar arrays.

Example 4.8: Use of Higher-Order Nulls in Planar Arrays

Consider a 6x6 array with uniformly-spaced omni-directional elements in an interference environment consisting of three jammers. The weight set was designed to pass a user from the broadside direction (i.e elevation = 90°) while suppressing interference using high-order null adaptive processing.

The jamming environment for this example is summarized in the following table:

Signal	Azimuth	Elevation	Freq.	JNR (dB)
Jammer 1	+10°	60°	1.0	40
Jammer 2	+70°	60°	1.0	40
Jammer 3	130°	60°	1.0	40

Table 4.4: Parameters for the Two-Dimensional Broadband Scenario of Example 4.8

Higher-order null processing was performed by utilizing the modified steady-state covariance matrix $\tilde{R}_{xx}$:

$$\tilde{R}_{xx} = R_{xx} + A_x R_{xx} A_x^H + A_y R_{xx} A_y^H \tag{4.65}$$

The matrix A_x was a diagonal matrix, the elements of which were determined so as to multiply the sensor data by a ramp in the x-direction of the array. Similarly, the matrix A_y was a diagonal matrix, the elements of which were determined to as to multiply the sensor data by a ramp in the y-direction of the array. The Wiener equation was solved using the modified covariance matrix and beampatterns for the various cuts in azimuth and elevation were computed. As discussed previously, this processing produces second-order nulls. Thus, we expect to see zero gain and flat gain derivative, both respect to azimuth and elevation, at the locations of the jammers.

Figure 4.18a depicts the beampattern of the array at the fixed elevation angle of 60° (corresponding to the elevation angle of all three jammers). Note that the array places nulls in azimuth on all three jammers. Figure 4.18b depicts a scaled version of the beampattern derivative with respect to azimuth. This derivative was formed by computing the first difference of the beampattern for closely spaced samples in azimuth. It is evident from the Figure 4.18 that high-order null processing is effective in placing both nulls and flat derivative in azimuth on all jammers present.

Figure 4.18: Beampattern and Beampattern Derivative for Planar Array With Higher-C der Null Processing - Elevation = 60° (Example 4.8)

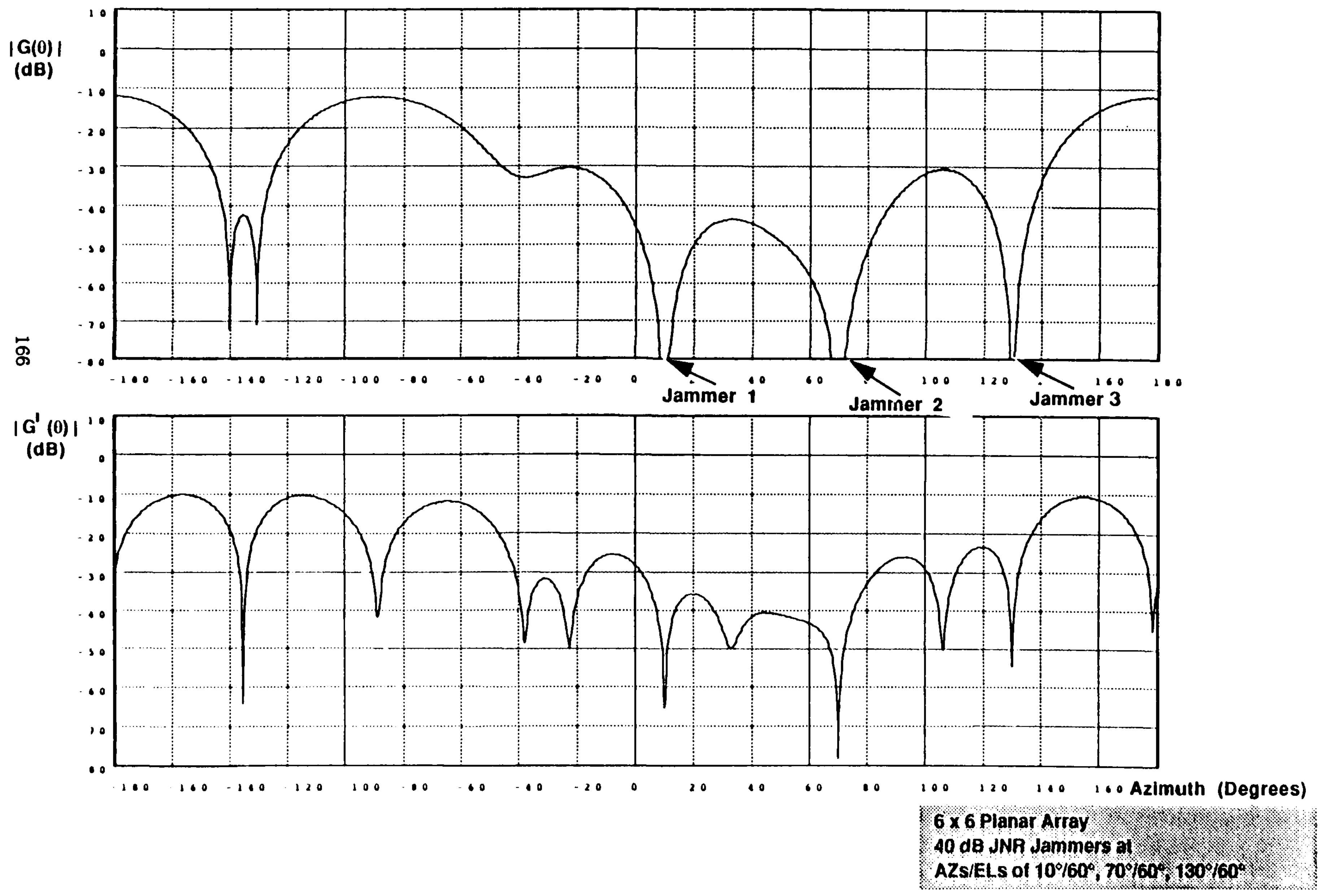


Figure 4.19 depicts the beampattern of the array as a function of elevation at each jammer azimuth. Curve 1 corresponds to the azimuth of Jammer 1, Curve 2 corresponds to the azimuth of Jammer 2, and Curve 3 corresponds to the azimuth of Jammer 3. All curves exhibit a deep null at the elevation angle of $\phi = 60°$. Figure 4.20 depicts the derivative of the beampattern with respect to elevation as a function of elevation. As can be seen the derivative also exhibits a null at the elevation angle of $\phi = 60°$ for all three jammer azimuths.

End of Example 4.8

4.2.7 Producing Clustered Nulls (Planar Arrays)

Clustered nulls can be achieved by replacing the A_x and A_y matrices by a diagonal matrix $T^{(i)}$. The elements of $T^{(i)}$ are chosen so as to ***construct a set of modulated counterparts of each snapshot*** so that it appears to the adaptive processor as having arrived from a set of nearby angles.

So see this more clearly, consider an arrival vector from azimuth angle θ and elevation angle ϕ. The component of this vector at the sensor with position x_m, y_n can be written as

$$u_{m,n}(\theta,\phi) = e^{j(\Omega_x x_m + \Omega_y y_n)} \tag{4.66}$$

Spatially modulating this vector is equivalent to multiplying the vector elements by a two-dimensional complex exponential of the form

$$T^{(i)}_{m,n} = e^{j(\delta_x x_m + \delta_y x_n)} \tag{4.67}$$

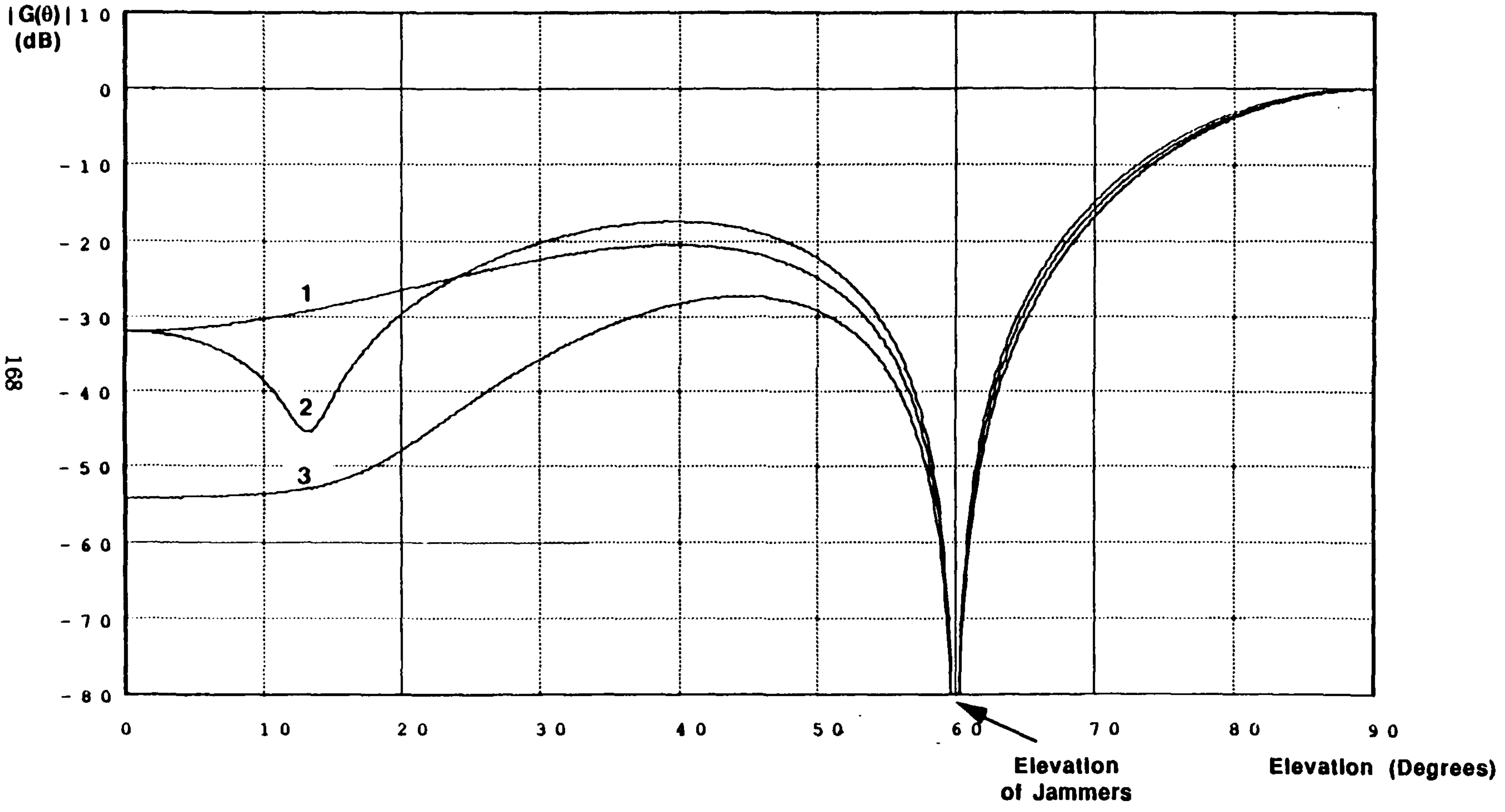
so that the component at the sensor x_m, y_n is given by

$$u_{m,n}(\theta,\phi) = e^{j(\Omega_x x_m + \Omega_y y_n)} * e^{j(\delta_x x_m + \delta_y x_n)} = e^{j([\Omega_x + \delta_x] x_m + [\Omega_y + \delta_y] y_n)} \tag{4.68}$$

The arrival vector is now translated in electrical angle coordinates Ω_x, Ω_y. If multiple jammers are present, modulation by the complex angle ***translates each jammer by a fixed offset in electrical angle.*** Note that the fixed translation in electrical angle coordinates Ω_x, Ω_y does not correspond to fixed translation in physical angle coordinates θ, ϕ.

Multiple $T^{(i)}$ matrices can be used so as to synthesize a series of snapshots corresponding to arrivals from multiple synthetic emitters ***near*** the actual emitters. In this case the adaptive processor will sense (in the high JNR case) arrivals coming from both the actual set of jammer directions as well as from the set of translated directions.

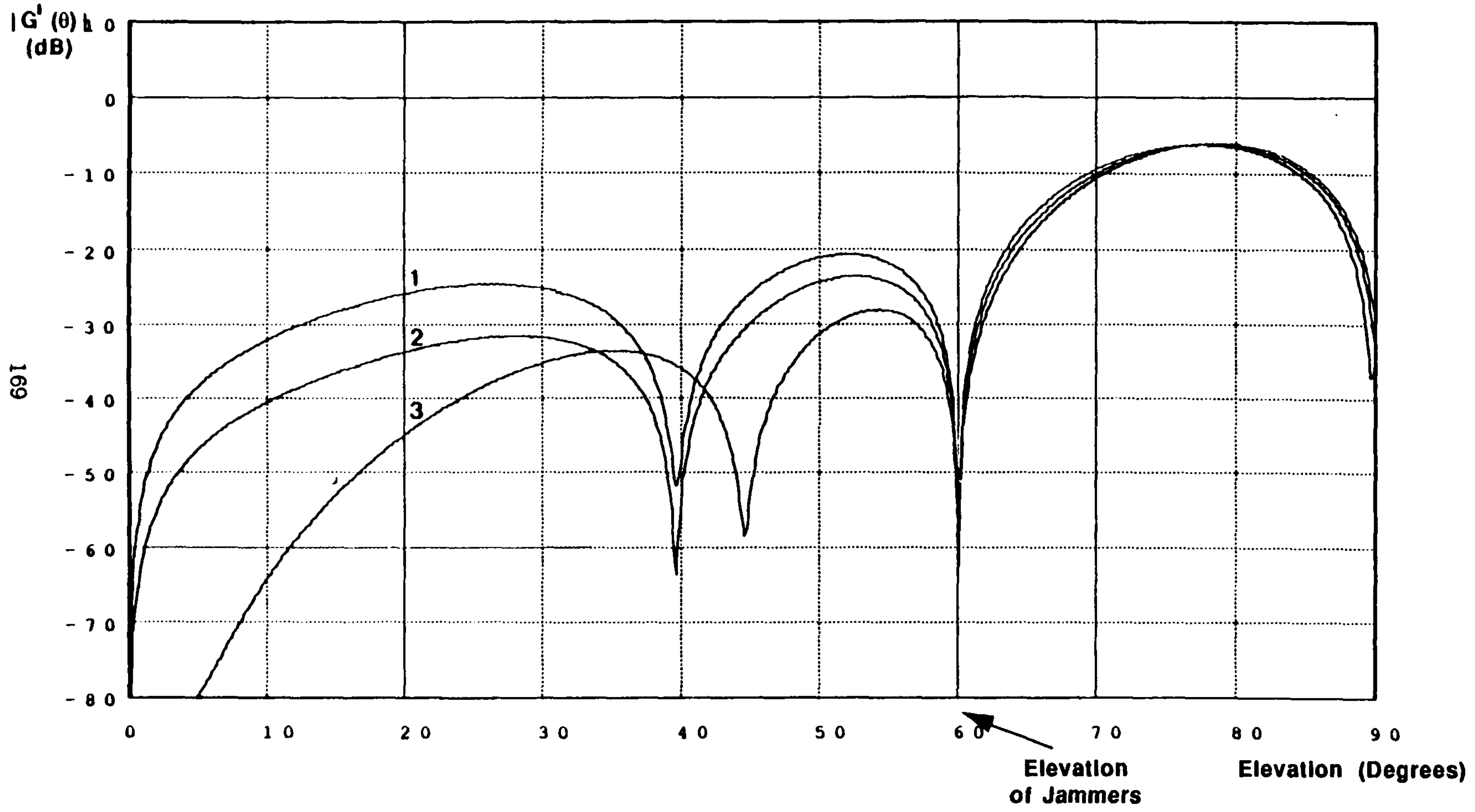
Figure 4.19: Beampattern for Planar Array With Higher-Order Null Processing (Example 4.8)



CURVES:
1. Beampattern at Jammer 1 Azimuth
2. Beampattern at Jammer 2 Azimuth
3. Beampattern at Jammer 3 Azimuth

6 x 6 Planar Array
40 dB JNR Jammers at
AZs/ELs of 10°/60°, 70°/60°, 130°/60°

Figure 4.20: Beampattern Derivative for Planar Array With Higher-Order Null Processing (Example 4.8)



CURVES:
1. Beampattern Derivative at Jammer 1 Azimuth
2. Beampattern Derivative at Jammer 2 Azimuth
3. Beampattern Derivative at Jammer 3 Azimuth

6 x 6 Planar Array
40 dB JNR Jammers at
AZs/ELs of 30°/60°, 70°/60°, 130°/60°

Similarly, by providing the adaptive algorithm with the modified covariance matrix described by

$$\tilde{R}_{xx} = \sum_{i=0}^{M-1} T^{(i)} \hat{R}_{xx} (T^{(i)})^H \tag{4.69}$$

it is possible to achieve M clustered nulls near each jammer, provided that sufficient degrees of freedom are available.

The following example illustrates the use of clustered-null processing in planar arrays.

Example 4.9: Use of Clustered Nulls in Planar Arrays

Consider a 7x7 planar array with uniformly-spaced omni-directional elements. The quiescent pattern for this example was chosen to be omni-directional so that the array is null-steering only. That is, the quiescent steering vector was selected to be all zeros except for unity gain at the sensor in the middle of the 7x7 array. The jamming environment for this example is summarized in the following table:

Signal	Azimuth	Elevation	Freq.	JNR (dB)
Jammer 1	+45°	60°	1.0	40
Jammer 2	+135°	60°	1.0	40

Table 4.5: Parameters for the Two-Dimensional Broadband Scenario of Example 4.9

Clustered-null processing was performed by modulating the covariance matrix with a 2D complex exponential function with parameters $\delta_x = \pi \cos 80° \sin 110°$ and $\delta_y = \pi \cos 80° \cos 110°$. A diagonal matrix $T^{(1)}$ was constructed and applied to the steady-state covariance matrix to compute a modified covariance matrix $\tilde{R}_{xx}$ given by

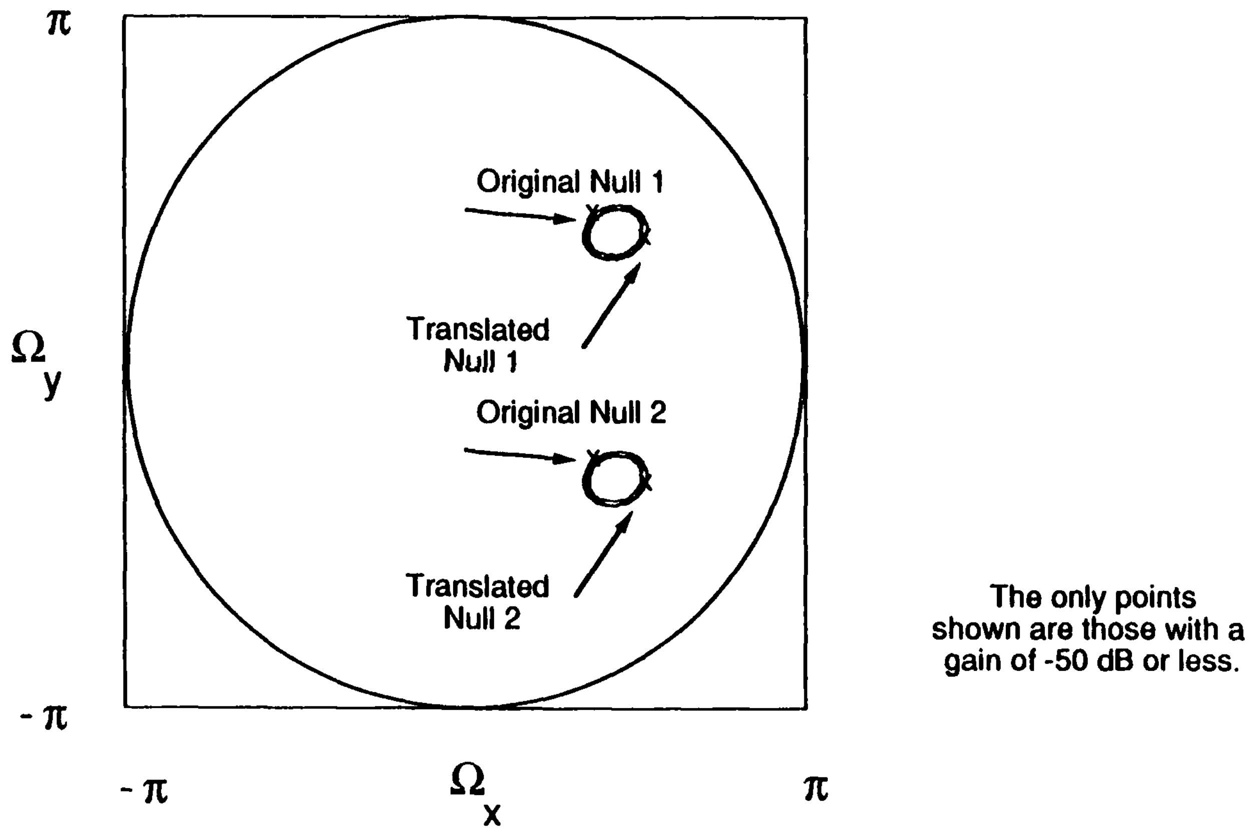
$$\tilde{R}_{xx} = R_{xx} + T^{(1)} R_{xx} (T^{(1)})^H \tag{4.70}$$

The modified covariance was then used in the Wiener equation to derive a set of weights. The beampattern of this weight set was then computed.

Figure 4.21 depicts the two-dimensional beampattern for this set of weights as a function of Ω_x and Ω_y. Only points in the pattern with gain less than -50 dB are contoured. Recall that electrical angles Ω_x and Ω_y are related to the physical angles θ and ϕ through the relationships.

$$\Omega_x = \pi \cos\phi \sin\theta \tag{4.71}$$

Figure 4.21: 2D Beampattern
for a Planar Array with Clustered-Null Processing.
(Example 4.9)



7x7 Planar Array.
40 dB JNR Jammers at Az/Els's of
45/60, 135/60

and

$$\Omega_y = \pi \cos\phi \cos\theta \tag{4.72}$$

The positions of both the original nulls, corresponding to the jammer locations, and the positions of the translated nulls are indicated in the figure. It can be seen that the both translated nulls are shifted by a fixed vector in the Ω_x, Ω_y plane corresponding to the values of δ_x and δ_y.

Also note that an entire contour of nulls is present. This underscores the fact that nulls for a planar array are contours, as opposed to nulls for a linear array which are points. In this particular case, the contours pass through both the locations of the jammers as well as through the translated locations. The shape of the contour is dependent on the choice of the quiescent steering vector - different choices of the quiescent vector will yield different contours all of which pass through the for locations corresponding to the original jammer locations and the translated locations. Additional nulls in the cluster could also be achieved by modulating the covariance matrix with additional matrices $T^{(i)}$.

End of Example 4.9

4.2.8 Producing Elevation Nulls in Planar Arrays

An antenna pattern $G(\theta, \phi)$ with higher-order nulls yields partial derivatives of $G(\theta, \phi)$ both with respect to θ and also ϕ equal to zero. It is evident from the discussion of Section 4.2.1 that requiring partial derivatives of $G(\theta, \phi)$, with respect to ϕ (elevation), to equal zero is useful with regard to providing extended bandwidth suppression of jammers. Specifically, the conditions assure that the pattern provides strong suppression over an extended sector of elevation, and thereby assures suppression as the pattern scans in elevation due to frequency change. However, the derivative conditions with respect to θ (azimuth) provide no comparable benefit. Thus, it is of interest to consider techniques that provide extended suppression in elevation, but not in azimuth. Such techniques provide extended bandwidth suppression of jammers while requiring half as many many degrees of freedom as higher-order null techniques. We have devised one such technique which we designate as the elevation null technique.

The technique is an extension of the ideas relating to higher-order nulls and clustered nulls for planar arrays. High-order nulls and clustered nulls were achieved by multiplying each sensor signal by an appropriate factor. Thus, each snapshot vector was multiplied by the appropriate diagonal matrix to produce $A_x\vec{x}(n)$, $A_y\vec{x}(n)$, or $T^{(i)}\vec{x}(n)$. We can generalize this notion by *operating* on each arrival vector with other linear operators denoted by the matrix L. For elevation nulls, L is an operator which approximately re-samples the snapshot at a different spatial rates.

To see this further, consider again an arrival vector from azimuth angle θ and elevation angle ϕ. The component of this vector at the sensor with position x_m, y_n can be written as

$$u_{m,n}(\theta,\phi) = e^{j[\Omega_x x_m + \Omega_y y_n]} = e^{j2\pi f/c \cos\phi[\sin\theta x_m + \cos\theta y_n]} \tag{4.73}$$

Note that an arrival coming from the same angle but at the frequency αf is expressed as

$$e^{j2\pi\alpha f/c\cos\phi[\sin\theta x_m + \cos\theta y_n]} = e^{j2\pi f/c\cos\phi[\sin\theta(\alpha x_m) + \cos\theta(\alpha y_n)]} \tag{4.74}$$

Thus, the arrival vector at lower frequency is related to the original arrival vector by a ***compression factor*** α ***in*** x ***and*** y. For an array with equally spaced elements spaced at half wavelength intervals corresponding to the highest frequency present, this compression can be written as follows

$$\tilde{u}_{m,n}(\theta,\phi) = u_{\alpha m,\alpha n}(\theta,\phi) = e^{j\pi\cos\phi[\sin\theta(\alpha m) + \cos\theta(\alpha n)]} \tag{4.75}$$

Thus, by operating on snapshots via a spatial compression (re-sampling) operation, it is possible to synthesize fictitious arrivals from the same location but at different frequencies. Spatially oversampling snapshots synthesizes fictitious arrivals from lower frequencies, while spatially undersampling yields fictitious arrival from higher frequencies. It is also possible to operate directly on the covariance matrix.

Numerous techniques for re-sampling two-dimensional signals are available in the digital signal processing literature. In our work, we have studied only the simplest of these methods as applied to constructing elevation nulls. In particular, we investigated re-sampling based on bi-linear interpolation of two-dimensional array snapshots (or the associated covariance matrix). Thus, to predict the value of the snapshot at $\alpha m, \alpha n$ we linearly interpolated from the four nearest neighbors. This technique is primarily applicable to oversampling, i.e. for $\alpha < 1.0$, corresponding to lower frequencies.

The following example illustrates the utility of elevation nulls in a planar array.

Example 4.10: Using Elevation Nulls to Extend the Bandwidth of a Planar Array

As discussed in Chapter 3, the presence of the user signal can have a severe impact on the performance of an adaptive array when the user signal is contained in the data used for adaptation. Chapters 3 and 5 describe this effect and candidate solutions, based on spatial pre-preprocessing. In this example, we present a method for reducing the impact of the user signal by ***measuring jamming data outside of the user frequency band and then extending the nulls in frequency using elevation null techniques.***

The scenario involves a 6x6 planar, narrowband array. The quiescent weight vector is selected to pass the user signal arriving from broadside. The interference consists of two $50dB$ JNR jammers at azimuth/elevation angles of $10°/70°$ and $100°/70°$. The user signal is also broadband (frequency hopped in a relatively broad frequency band) and the omnidirectional array elements are uniformly spaced at half-wavelength intervals corresponding to the highest user frequency present. We assume that the adaptive nulling system does not have explicit knowledge of the exact user hop frequency but that it knows the band within which the user is operating. It is also assumed that the ***broadband jammers transmit over a broad band which is larger than the user operating band.*** The essence of the algorithm is to measure the out-of-band jammer energy in order to establish a set of adaptive array weights valid across the entire band.

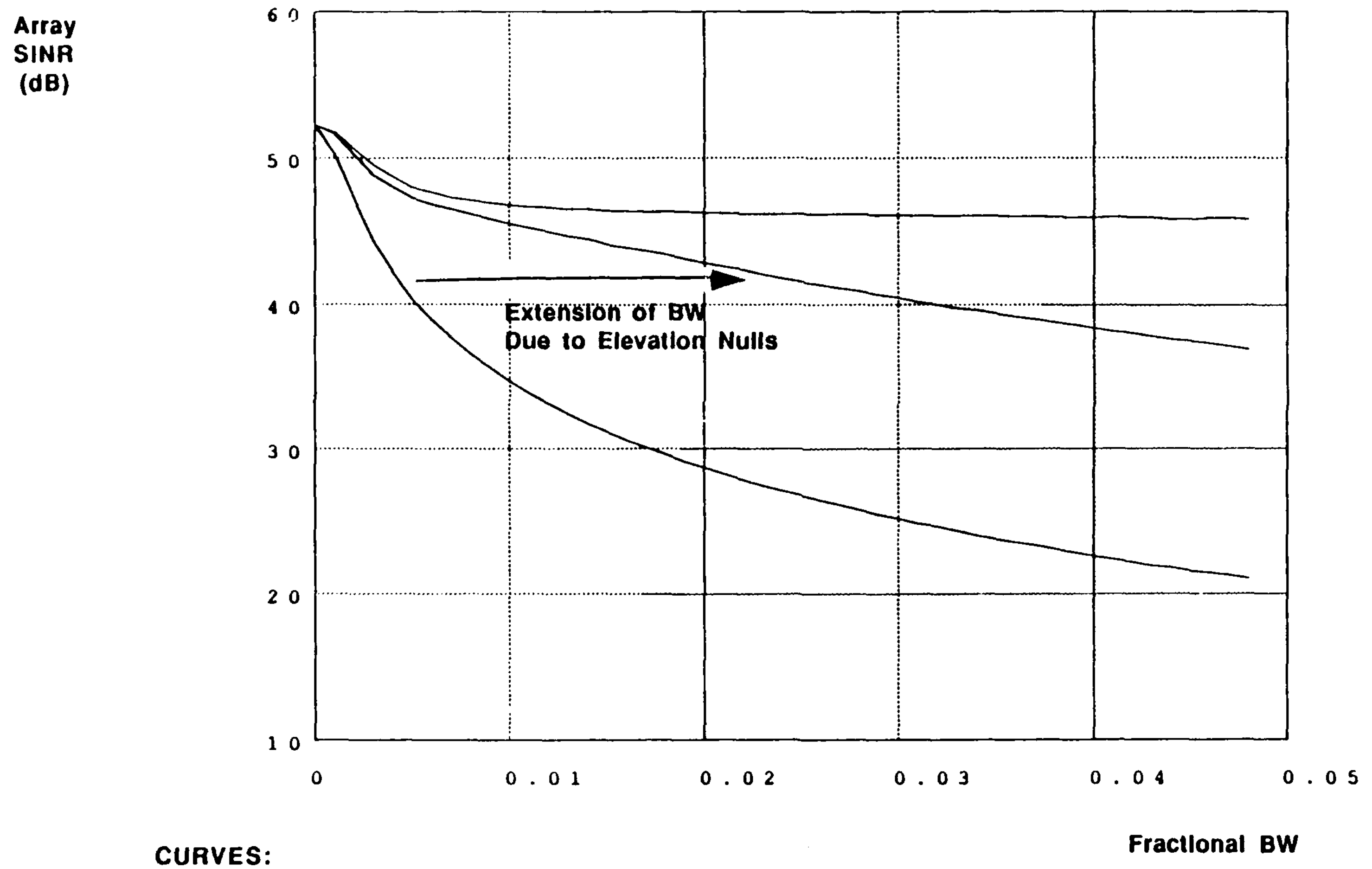
If a spatial covariance matrix is formed from out-of-band jamming energy, then the resultant set of weights will provide nulls which are appropriate only for the measurement band. The objective of elevation nulls is to provide a set of weights which are applicable at other frequencies within the adjacent user band.

Figure 4.22 illustrates the SINR performance as a function of the operating bandwidth for several adaptation schemes. In each case the nulling weights were calculated by solving the (modified) Wiener-Hopf equation. The schemes differ according to the frequency band over which the actual covariance matrix R_{xx} was computed, and also whether or not a modified covariance matrix $\tilde{R}_{xx}$ was utilized.

Curve 1 in Figure 4.22 is a baseline curve which depicts the SINR when ***the jamming energy is measured over the entire user operating band.*** This baseline curve might correspond, for example, to a cooperative user scenario, where the array can measure only the interference, without user signal presence, over the entire user band. Curve 2 depicts the SINR when the ***jamming energy is measured in a narrow frequency band slightly higher in frequency than the upper edge of the operating band.*** The SINR falls off quite rapidly as the jammer bandwidth increases. This effect is due to the fact that the nulls formed from the narrowband measurement data are of limited utility over larger operating bandwidths.

Elevation nulls were achieved by bi-linearly interpolating the covariance matrix for the narrowband interference data and then adding this matrix to the measured covariance ma-

Figure 4.22: Use of Elevation Nulls in a Planar Array to Extend Operating Bandwidth (Example 4.10)



CURVES:
1. Baseline Performance Using Full Band Interference-Only Measurements
2. Performance Using Out-of-Band Interference Measurements
3. Performance Using Out-of-Band Interference Measurements and Elevation Nulls

6 x 6 Planar Array Sterred Broadside
40 dB JNR Fractional BW Jammers at
AZs/ELs of: 10°/70°, -20°/80°
40 dB SNR Frequency-Hopped User

trix. Curve 3 in Figure 4.22 depicts the result of this processing. Clearly, the performance of the array improves substantially. The operating bandwidth corresponding to an SINR which is down 10 dB from the narrowband SINR is extended by more than a factor of four due to the use of elevation nulls. The performance improvement is due to the fact that ***elevation nulls extend the bandwidth of the nulls computed from narrowband measurement data.*** In comparing Curve 3 with Curve 1 (the baseline curve) we see that further improvement is possible. It is felt that better re-sampling schemes applied to the covariance matrix would yield this improved performance.

End of Example 4.10

4.3 Broadband Arrays

4.3.1 Importance Of Linear Constraints For Broadband Arrays

It is desirable to impose linear constraints on the weights of a broadband array for the same reasons as in the case of a narrowband array. Multiple constraints can be used to maintain specified gain in both the spatial and spectral response of the broadband array. Thus, constraints can be used to place nulls over specific bands in specific directions when jammer characteristics are known a priori. Similarly, multiple constraints are effective in providing fixed gain to one or more users over different operating bands. Multiple constraints also provide a means for reduncing the dimensionality of the problem, thereby reducing the computation required[9].

Multiple linear constraints can be incorporated into the optimal stacked weight solution as follows. The output power p of a broadband array is given in terms of the stacked weight set $\vec{w}$ and the stacked covariance matrix R_{xx} as

$$p = \vec{w}^H R_{xx} \vec{w} \tag{4.76}$$

A set of M linear constraints on the weight set can be used to provide steering control via the constraint equation

$$C^H \vec{w} = \vec{f} \tag{4.77}$$

The optimal stacked weight set which minimizes the output power of the broadband array subject to this set of constraints is given by

$$\vec{w}_o = R_{xx}^{-1} C (C^H R_{xx}^{-1} C)^{-1} \vec{f} \tag{4.78}$$

The following examples illustrate the use of constraints in multiple broadband arrays.

Example 4.11: Utilization of Multiple Constraints in a Linear Broadband Array

Consider a broadband 1D array with 16 sensors and 8 taps per sensor, and assume that the direction (azimuth) and normalized frequency band of a user are known to be 10 degrees, and 0.75-1.0 Hz respectively. Constraints were selected to provide unity magnitude at 4 evenly spaced frequencies across the normalized frequency band 0.75-1.0, in the 10 degree look direction. The phase constraints were chosen so that first differences of the unwrapped phase, between adjacent frequency samples, yield a group delay of 4 samples. Thus, a total of 4 degrees of freedom were used to achieve a quiescent broadband array design which passes the fractional bandwidth user. The remaining 124 degrees of freedom are available to an adaptive algorithm for nulling jammers.

These 4 constraint conditions were used to construct the minimum-norm weight set given by

$$\vec{w}_m = C(C^H C)^{-1}\vec{f} \tag{4.79}$$

The weights were then used to construct beampatterns depicting the response of the array. Figure 4.23 shows the response of the broadband array as a function of angle for a normalized frequency of 0.9, which is within the user passband. It is apparent from the figure that the gain of the array is unity in the look direction of the user. Essentially, the array points toward the user at this, and all frequencies within the frequency band 0.75-1.0.

The frequency response of the array in the user look direction is shown in Figure 4.24. From this figure, it can be seen that, over the user frequency band, the magnitude of the response of the array is unity and the group delay is 4 samples. Note that the selection of only 4 constraints spaced evenly over the fractional band was sufficient to achieve the desired response over the entire fractional band. Also note that the out-of-band gain of the array is quite small, thus providing minimal noise power at the output of the array.

End of Example 4.11

Example 4.12: Utilization of Multiple Constraints in a Planar Broadband Array

Consider a 6x6 2D broadband array which has 8 taps per sensor. Multiple constraints were used to provide broadband gain in multiple directions. The 2D broadband array was designed to pass two fractional bandwidth users, and to null a fractional bandwidth jammer, whose position is assumed to be known. The characteristics of the 3 emitters are summarized in the following table

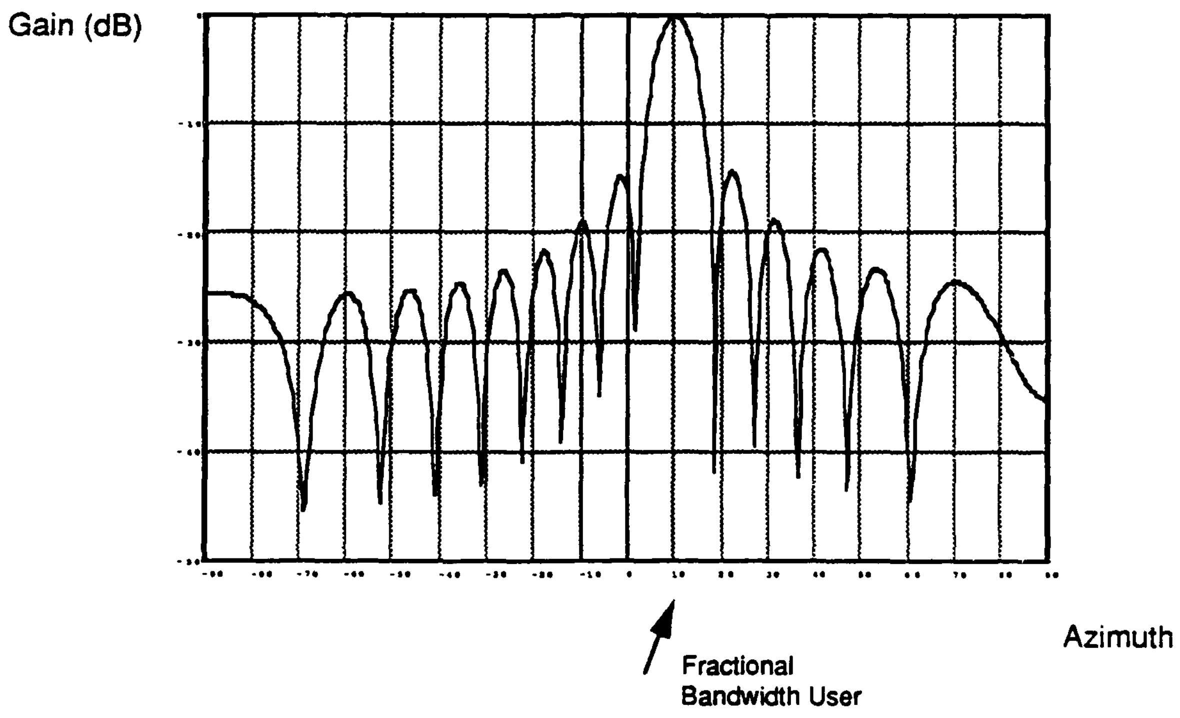


Figure 4.23

Beampattern (at f=0.9) for a Linear Broadband Array

(Example 4.11)

16 Element broadband array
with 8 taps per element. 4 constraints
were used to steer the array
toward a 25% fractional bandwidth
user at 10 degrees.

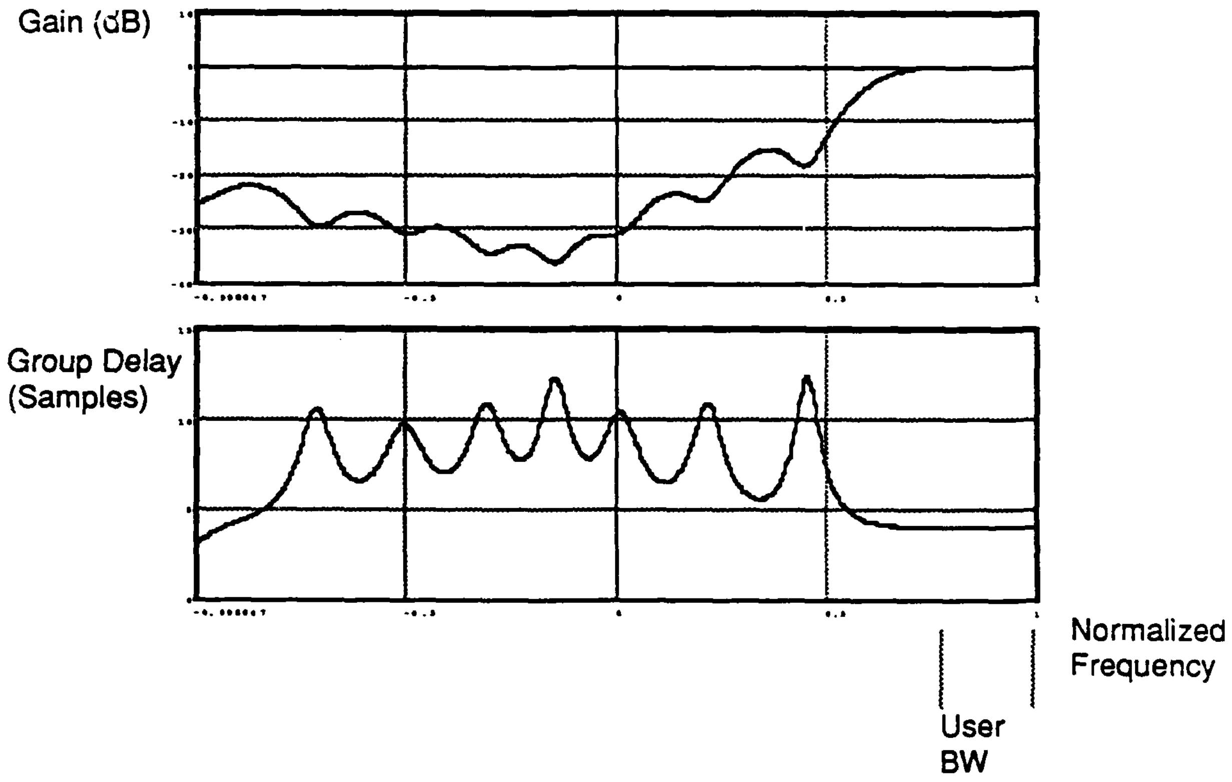


Figure 4.24

Frequency response of a Linear Broadband Array in the User Look Direction.

(Example 4.11)

16 Element broadband array
with 8 taps per element. 4 constraints
were used to steer the array
toward a 25% fractional bandwidth
user at 10 degrees.

	Azimuth	Elevation	Frac. BW
Emitter A (User)	10	70	0.75-1.0
Emitter B (User)	100	70	0.75-1.0
Emitter C (Jammer)	-110	70	0.75-1.0

Table 4.6: Emitter Characteristics for Example 4.12

The constraint conditions for this problem were selected so as to pass the two fractional bandwidth users with unity gain and group delay of 4 samples, and to null the fractional bandwidth jammer. To do this, 4 constraints, spread evenly across the fractional bandwidth, were selected for each of the 3 look directions.

Figure 4.25 shows the response of the array as a function of azimuth for the fixed elevation of 70 degrees and for the fixed frequency of 0.9. This frequency is within the common fractional bandwidth of the 3 emitters. Note that the array exhibits unity gain at the 2 users azimuths of 10 and 100 degrees, as well as a sharp null at the jammer azimuth of -110 degrees. Figure 4.26 shows the response of the array as a function of elevation for azimuths of 10, 100 and -110 degrees, and for the fixed frequency of 0.9. These curves again show unity gain at an elevation of 70 degrees for the user azimuths and a sharp null at 70 degrees for the jammer azimuth. Also note the sharp null in the elevation coordinate, in contrast to the distributed elevation null characteristics of narrowband arrays. Figure 4.27 depicts the magnitude of the frequency response in the look directions of the two users and the jammer. From this figure, it can be seen that flat magnitude is achieved across the fractional bandwidth of the two users, and that effective suppression of the fractional bandwidth jammer is also achieved. Finally, in Figure 4.28, it can be seen that the group delay of the array, in the look directions of the two users, is also flat across the fractional bandwidth.

This example demonstrates that multiple constraints can be used to provide broadband steering and nulling for a 2D array when the positions of all emitters are known a priori. In this particular example, a total of 12 degrees of freedom out of a possible 288 were expended, leaving the remaining 276 degrees of freedom available to an adaptive nulling algorithm.

End of Example 4.12

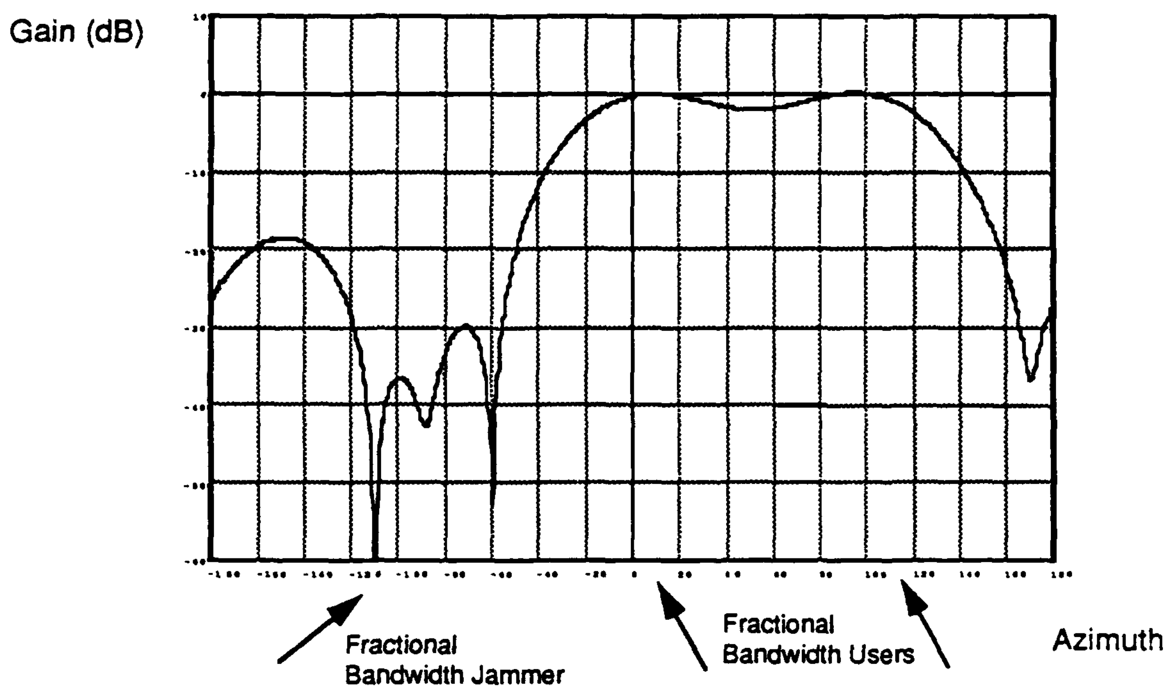


Figure 4.25

Beampattern (at f=0.9, elevation=70) for a Planar Broadband Array

(Example 4.12)

6x6 broadband array
with 8 taps per element. 8 constraints
were used to steer the array
toward a 25% fractional bandwidth
user at 10 degrees azimuth and
toward a 25% fractional bandwidth
user at 100 degrees azimuth.
4 additional constraints were
used to null the fractional bandwidth
jammer at -110 degrees. *All* emitters
were at an elevation of 70 degrees.

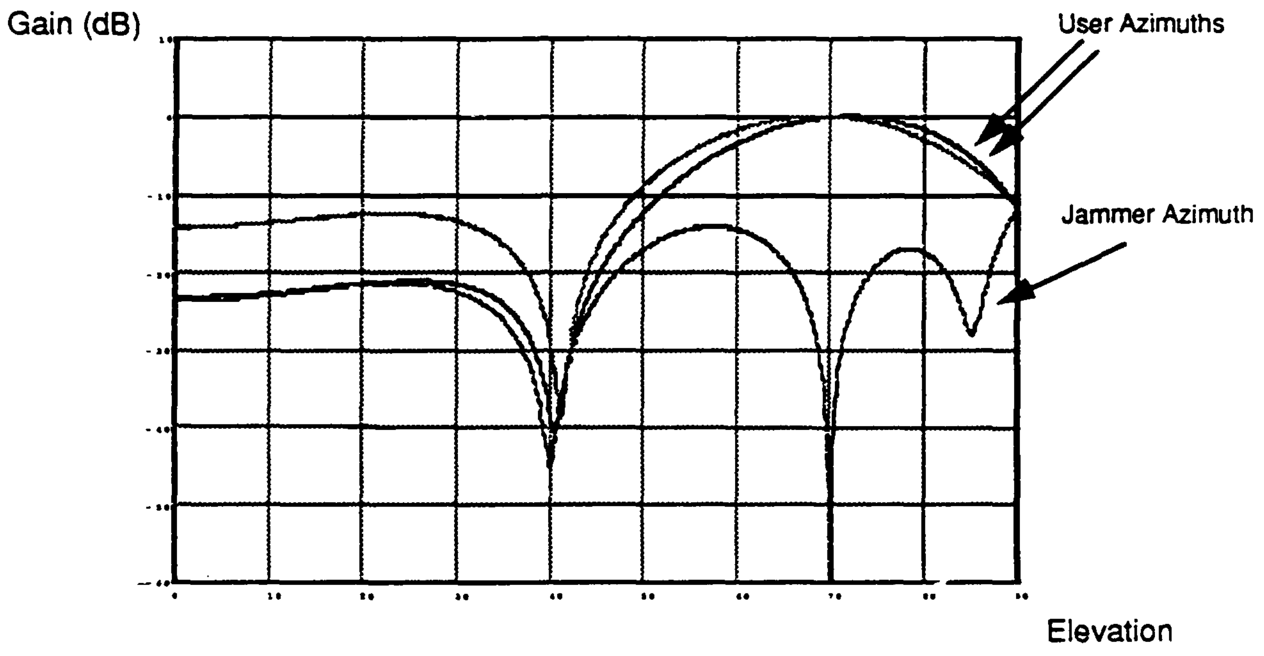


Figure 4.26

Beampattern (at f=0.9) for a Planar Broadband Array

(Example 4.12)

6x6 broadband array
with 8 taps per element. 8 constraints were used to steer the array toward a 25% fractional bandwidth user at 10 degrees azimuth and toward a 25% fractional bandwidth user at 100 degrees azimuth.
4 additional constraints were used to null the fractional bandwidth jammer at -110 degrees. All emitters were at an elevation of 70 degrees.

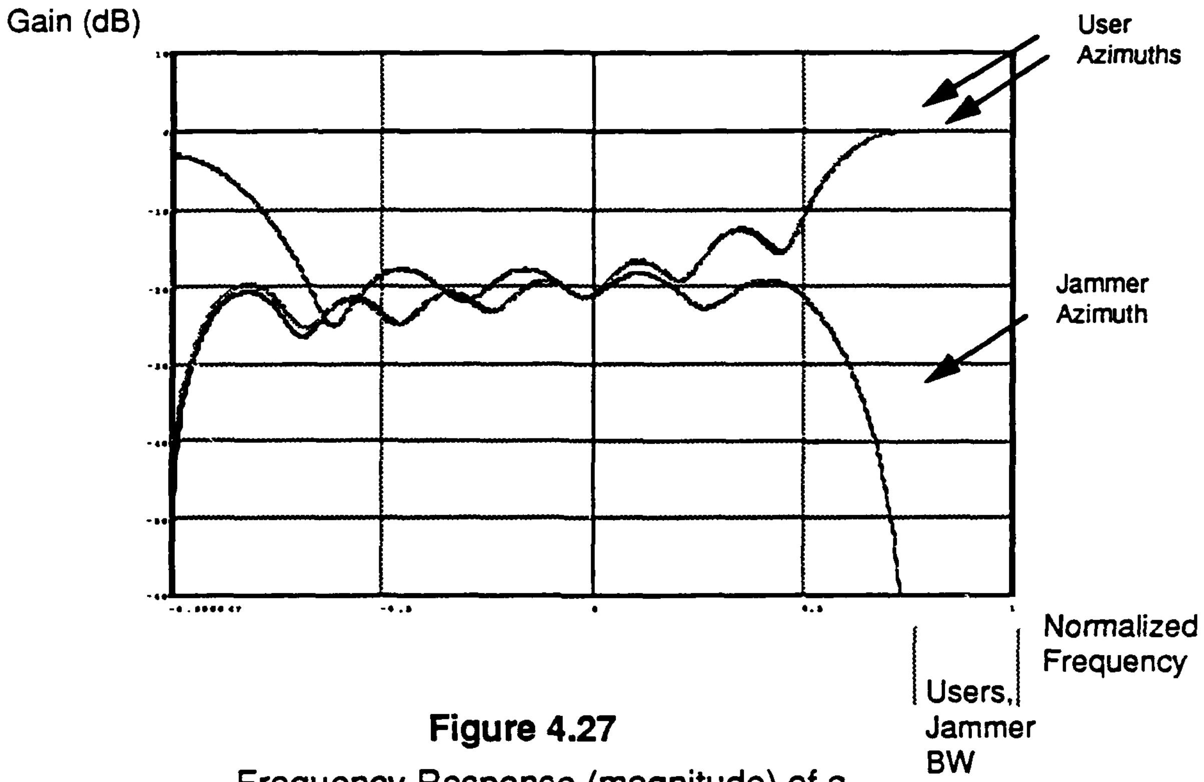


Figure 4.27

Frequency Response (magnitude) of a Planar Broadband Array at Elevation=70 Degrees.

(Example 4.12)

6x6 broadband array
with 8 taps per element. 8 constraints
were used to steer the array
toward a 25% fractional bandwidth
user at 10 degrees azimuth and
toward a 25% fractional bandwidth
user at 100 degrees azimuth.
4 additional constraints were
used to null the fractional bandwidth
jammer at -110 degrees. All emitters
were at an elevation of 70 degrees.

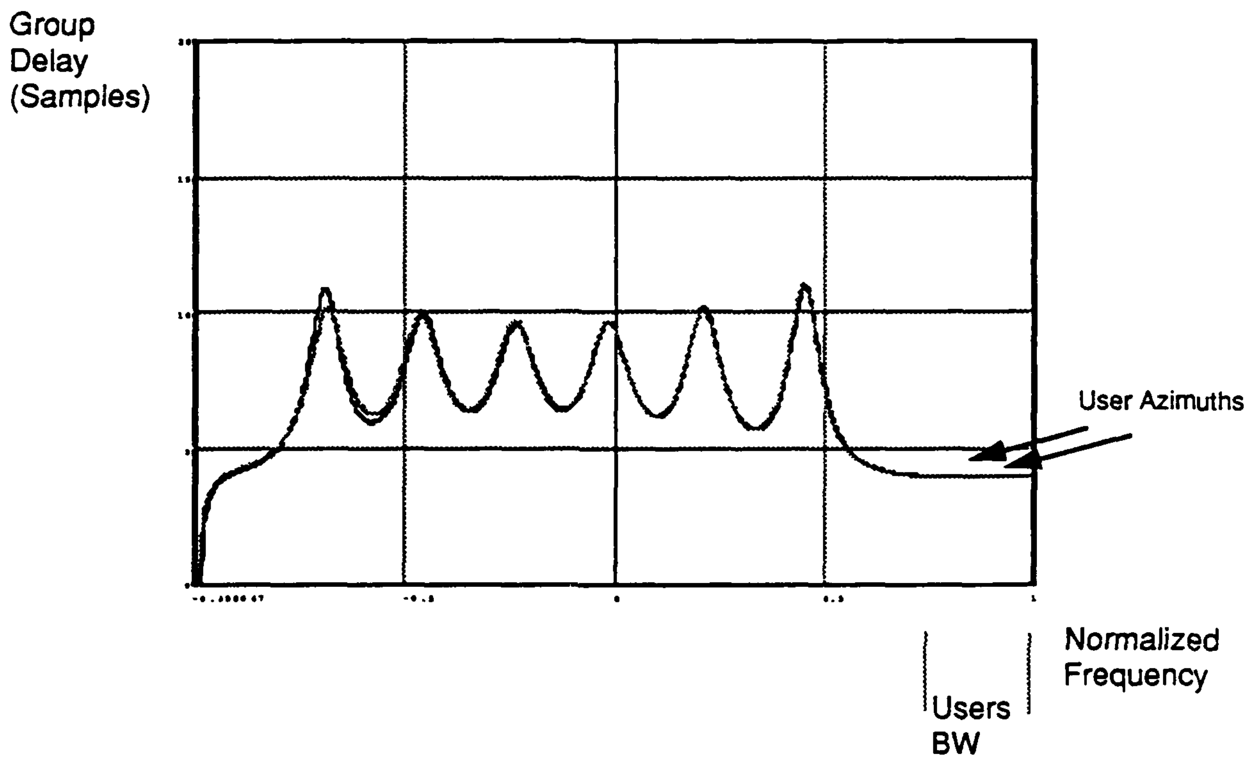


Figure 4.28

Group Delay of a Planar Broadband Array at Elevation=70 Degrees.

(Example 4.12)

6x6 broadband array
with 8 taps per element. 8 constraints
were used to steer the array
toward a 25% fractional bandwidth
user at 10 degrees azimuth and
toward a 25% fractional bandwidth
user at 100 degrees azimuth.
4 additional constraints were
used to null the fractional bandwidth
jammer at -110 degrees. All emitters
were at an elevation of 70 degrees.

4.3.2 Performance Of Optimal Broadband Arrays

The performance of broadband arrays differs from that of narrowband arrays in broadband environments, in terms of the antenna patterns of the adapted array. Specifically in the case of narrowband arrays, broadband suppression is achieved via extended angular suppression so that as the antenna pattern scans with frequency, suppression is achieved across a wide band of frequencies. By contrast, in the case of a broadband array, the spatial suppression of a broadband jammer is limited to a single direction. As a result, broadband arrays are better suited to suppressing jammers that are close to a user.

The improved angular resolution of a broadband array can best be understood by assuming the filters of the filter-and-sum processor to be realized as banks of narrowband filters as described in Chapter 3. Specifically, if the weights applied to the narrowband filters at a given center frequency are selected so as to pass the inband user signal and to optimally suppress the inband jammer signals, then the resultant weight set will produce sharp inband nulls in all angular coordinates. But, since this is the case for all the sets of bandpass filters at all center frequencies, the aggregate set of weights produces sharp angular nulls across the composite user frequency band. Finally, since the tapped delay line filter realization is (approximately) equivalent to the filter bank realization, it follows that optimal broadband arrays employing tapped delay line filters also must produce sharp angular nulls.

The following examples illustrate the use of optimal broadband arrays.

Example 4.13: Suppression of Jammers Using an Optimal Broadband Linear Array

Consider a 16 element linear broadband array with 8 taps per sensor. The constraint conditions were selected so as to provide unity gain and group de'ay of 4 samples in -5 degree look direction, at 4 evenly-spaced frequencies over the normalized fractional bandwidth of 0.75-1.0 Hz. Thus, 4 degrees of freedom were used to constrain the gain to a fractional bandwidth user at -5 degrees, and the remaining 124 degrees of freedom are available for adaptive nulling.

The simulated jamming environment consisted of a single 0 dB fractional bandwidth jammer, located at an azimuth of 20 degrees, with a normalized fractional bandwidth of 0.75-1.0 Hz. A jammer-to-noise ratio (JNR) of 40 dB has been assumed for this example.

The optimal weight set for this broadband beamformer was determined by computing the steady-state stacked covariance matrix for the interference environment and then using equation (4.78) to determine the weight vector. Note that in an adaptive broadband system, the stacked covariance matrix is unavailable and must be estimated, while in this s:mulation, the matrix is analytically determined using jammer direction and bandwidth information.

Figure 4.29 shows the frequency response of the array, after adaptation, in the user look direction of -5 degrees. Note that the magnitude and group delay are relatively flat over the constrained frequency band. Thus, the user will be passed with minimal distortion. Figure 4.30 shows the magnitude of the frequency response in the jammer look direction. As can be seen from the figure, the adaptation process has reduced the magnitude of the response over the frequency band of the jammer by more than 30 dB. Finally, Figure 4.31 shows the array output power integrated over the normalized frequency band of 0.75-1.0 Hz, plotted versus azimuth. This pattern is referred to as the *broadband beampattern.* It can be seen that the gain to the user is preserved, while a deep, narrow null has been placed on th broadband jammer.

End of Example 4.13

Example 4.14: Comparison Between Broadband and Narrowband Linear Arrays

Consider a 16 element linear array with 8 taps per sensor, which has an omni-directional quiescent pattern. That is, assume that the array is used to provide null- steering and that the user directions are unknown. The broadband interference environment has been chosen as a 0 dB monochromatic jammer with a normalized frequency of 1.0, located at an azimuth of -30 degrees, and a 0 dB fractional bandwidth jammer, extending over the normalized frequency band from 0.75 to 1.0, located at an azimuth of +30 degrees. A JNR of 40 dB has been assumed for this example.

This identical scenario was considered in Example 4.1 except that the 16-element linear array was narrowband. We will now compare the difference in performance between the optimal 16-element narrowband array and the optimal 16 element-broadband array in the presence of broadband interference.

To do this, the optimal weight set for the broadband array was computed, using Equation (4.78). The beampattern, as a function of azimuth for the fixed frequency of 1.0, is shown in Figure 4.32a. For reference, the beampattern for the optimal narrowband array is shown in Figure 4.32b. By comparing these two beampatterns, it can be seen that both the optimal narrowband and optimal broadband arrays provide a sharp directional null in the direction of the narrowband jammer at -30 degrees. However, while the narrowband array provides a broad null over a large range of azimuth near the fractional bandwidth jammer,

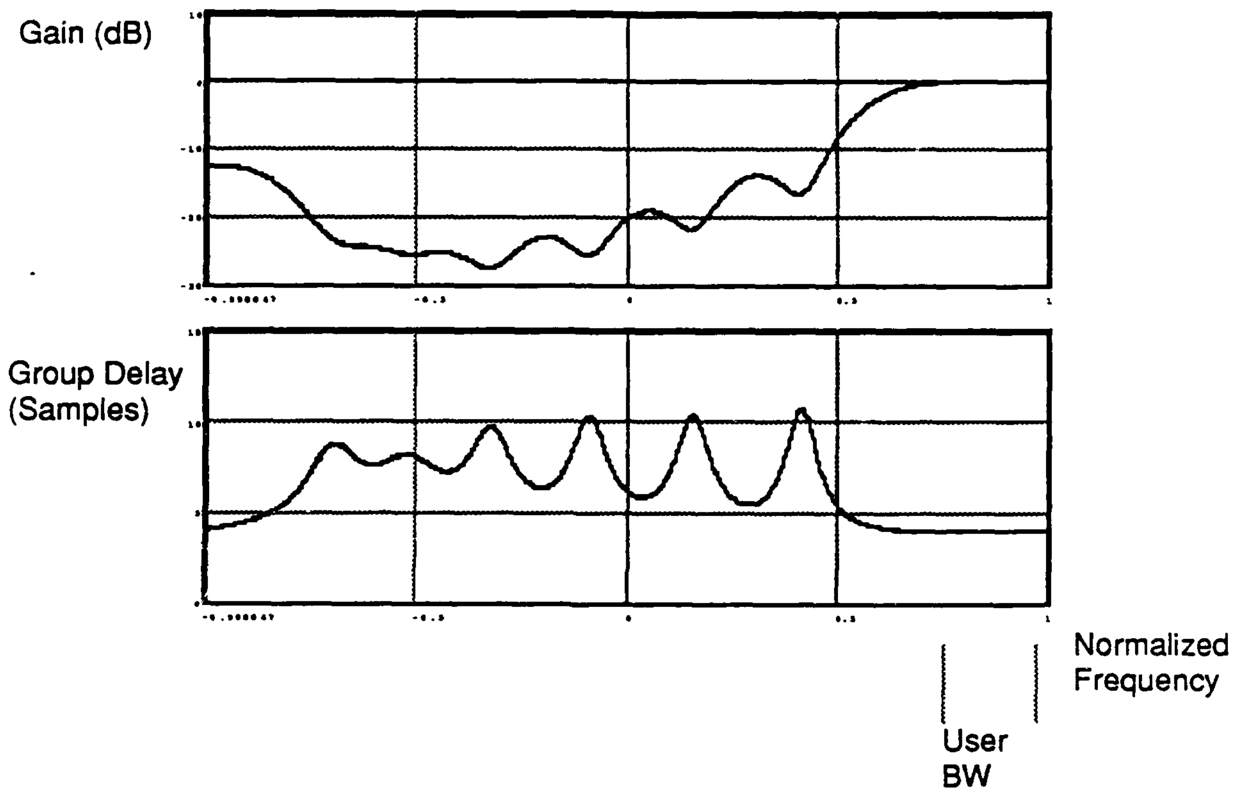


Figure 4.29

Frequency response of an Optimal Linear Broadband Array in the User Look Direction.

(Example 4.13)

16 Element broadband array with 8 taps per element. 4 constraints were used to steer the array toward a 25% fractional bandwidth user at -5 degrees. 0dB jammer with 25% fractional bandwidth was located at 20 degrees. JNR=40 dB

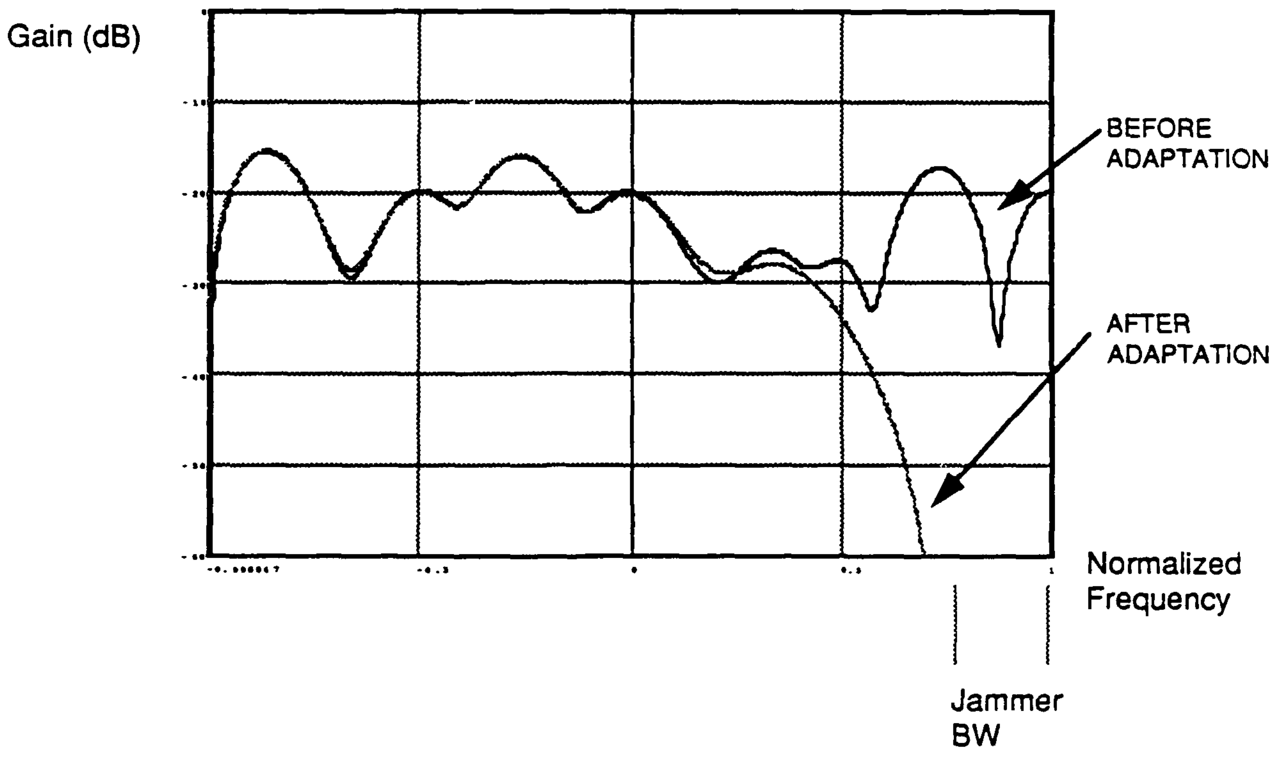


Figure 4.30

Frequency response (Magnitude) of an Optimal Linear Broadband Array in the Jammer Look Direction.

(Example 4.13)

16 Element broadband array
with 8 taps per element. 4 constraints
were used to steer the array
toward a 25% fractional bandwidth
user at -5 degrees. 0dB jammer
with 25% fractional bandwidth was
located at 20 degrees. JNR=40 dB

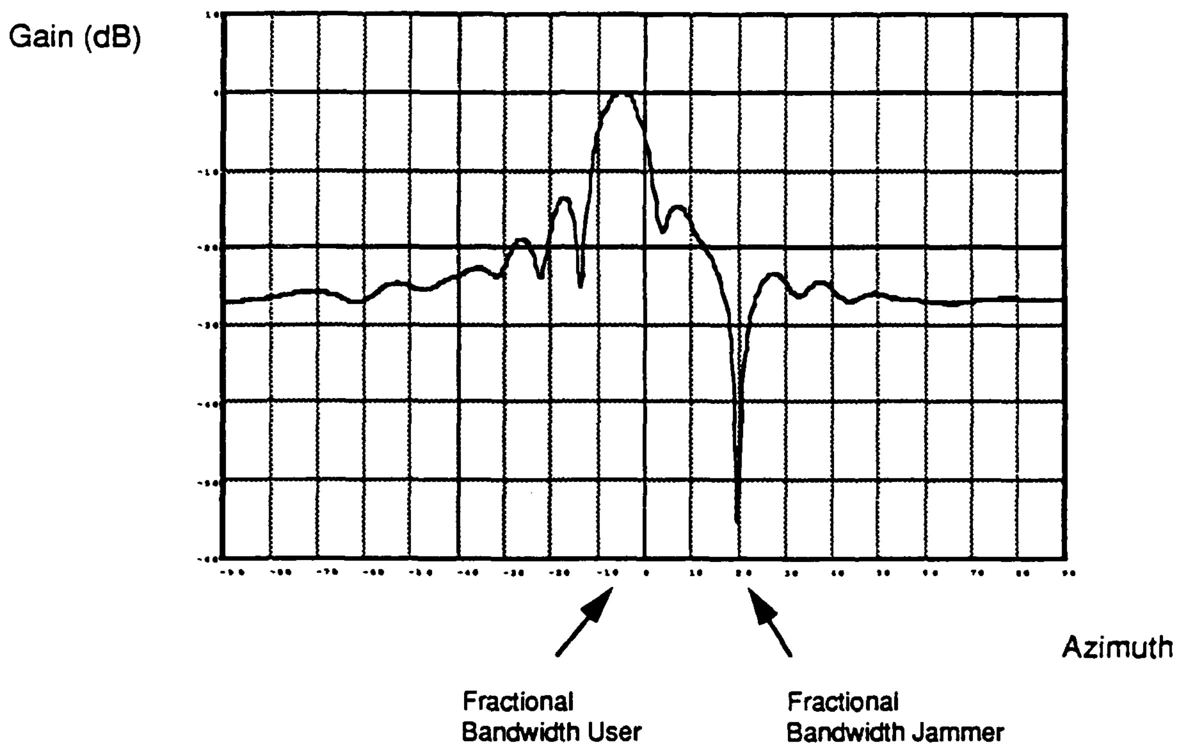


Figure 4.31

Broadband Beampattern (f=0.75-1.0) for an Optimal Linear Broadband Array

(Example 4.13)

16 Element broadband array with 8 taps per element. 4 constraints were used to steer the array toward a 25% fractional bandwidth user at -5 degrees. 0dB jammer with 25% fractional bandwidth was located at 20 degrees. JNR=40 dB

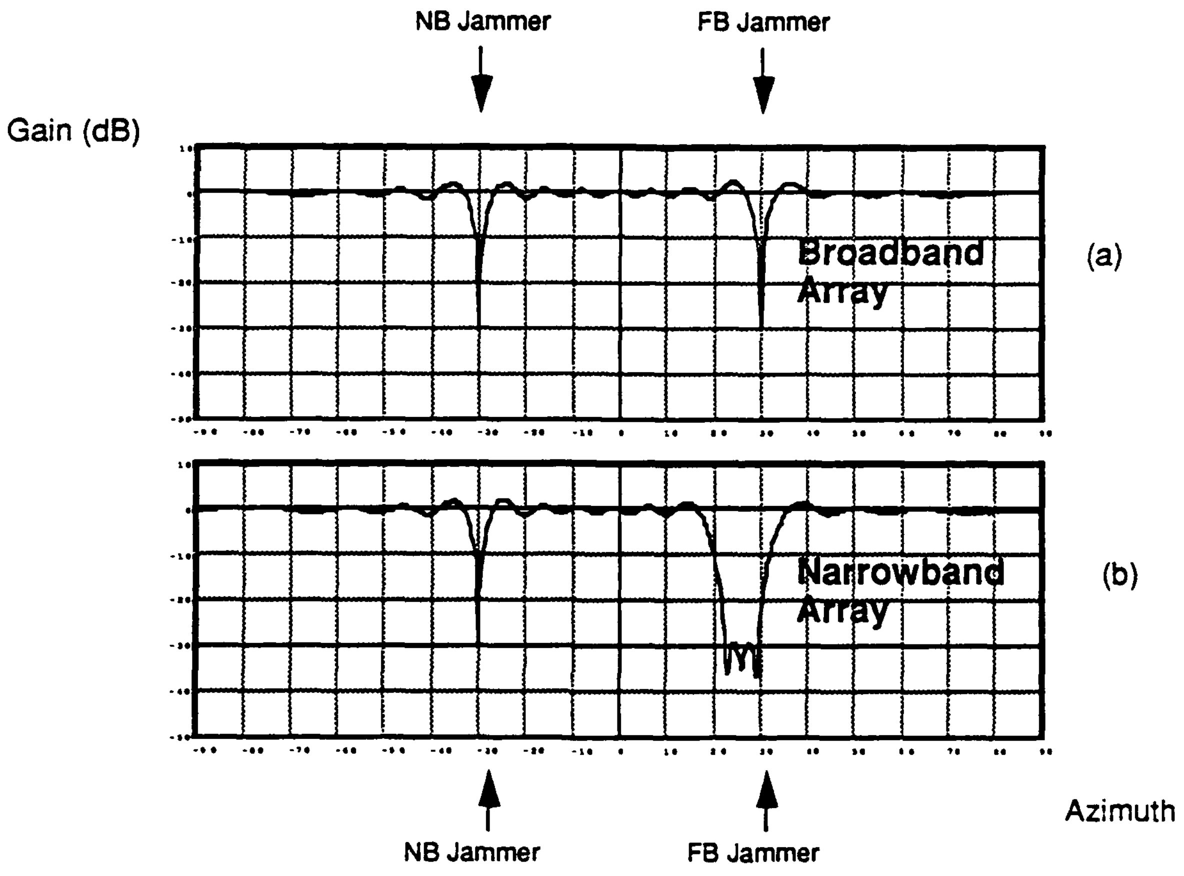


Figure 4.32

Beampatterns (at f=1.0) for Optimal Linear Broadband and Narrowband Arrays.

(Example 4.14)

16 Element broadband (8 taps per sensor) and narrowband arrays with omni-directional quiescent patterns. 0 dB narrowband jammer present at -30 degrees. 0 dB 25% fractional bandwidth jammer present at +30 degrees. JNR=40dB.

the broadband array provides a sharp null. Thus, the broadband array is able to excise the fractional bandwidth jammer, while the narrowband array must suppress this jammer by placing nulls over a large angular extent.

End of Example 4.14

Example 4.15: Comparison Between Broadband and Narrowband Planar Arrays

Consider a 6x6 planar broadband array with 8 taps per sensor which has an omnidirectional quiescent pattern. That is, assume that the array is used to provide null-steering and that the user directions are unknown. For this example, an interference environment has been chosen which consists of three jammers with identical powers, with differing fractional bandwidths and locations. The characteristics of the three jammers are summarized in the following table.

Jammer	Azimuth	Elevation	Power (dB)	Norm. Bandwidth
A	30	60	0	1.0-1.0
B	140	60	0	0.95-1.0
C	280	60	0	0.7-1.0

Table 4.7: Jammer Characteristics for Example 4.15

We now compare the difference in performance between the optimal planar broadband array and the optimal planar narrowband array in the presence of broadband interference. To do this, the optimal weight set for the broadband array was computed, using (4.78). The beampattern, as a function of azimuth for the fixed frequency of 1.0 and at an elevation of 60 degrees, is shown in Figure 4.33a. For reference, the beampattern for the optimal narrowband array is shown in Figure 4.33b. In comparing these two beampatterns, it can be seen that both the optimal narrowband and optimal broadband arrays provide sharp nulls in azimuth toward narrowband and fractional bandwidth jammers. However, Figure 4.34 depicts the differences in the beampatterns as a function of elevation. In Figure 4.34a is shown the broadband array response, at a normalized frequency of 1.0, as a function of elevation for each of the 3 jammer azimuths. Figure 4.34b shows the response for the optimal narrowband array. As can be seen from this figure, the effect of the additional taps in the planar broadband array is to greater sharpen the spatial extent of the nulls in elevation.

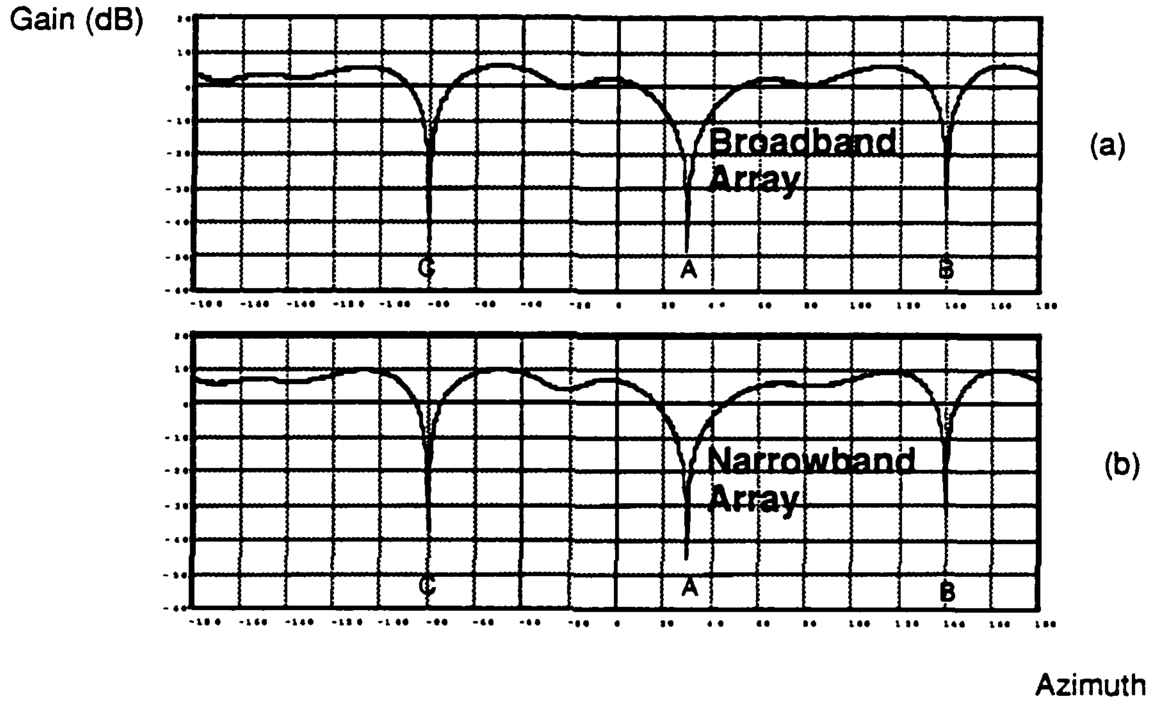


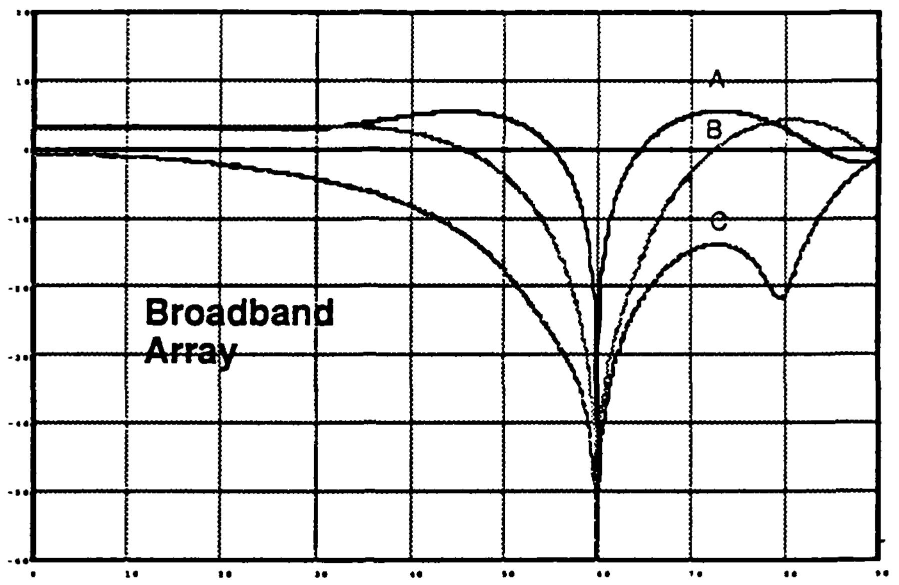
Figure 4.33

Beampatterns (at f=1.0, elevation=60) for Optimal Planar Broadband and Narrowband Arrays.

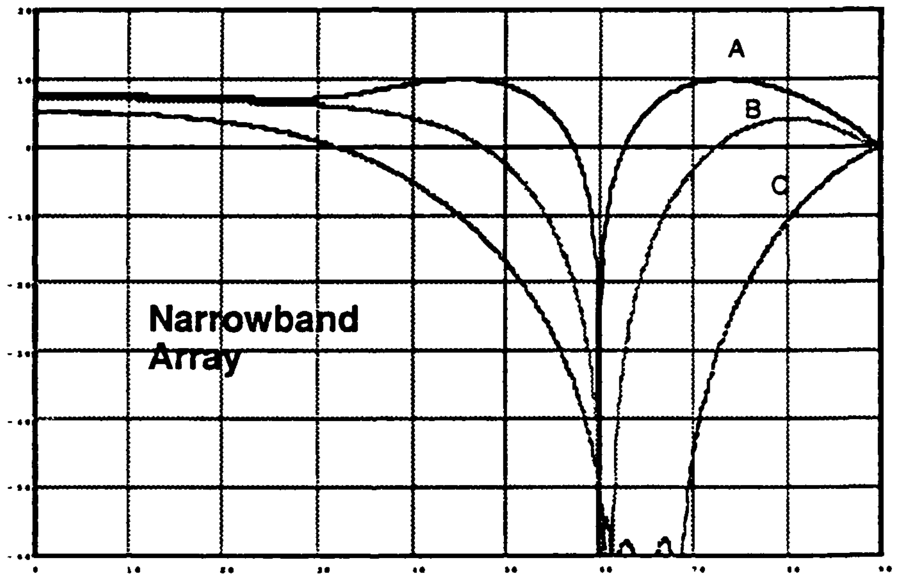
(Example 4.15)

6x6 2D narrowband array with omni-directional quiescent pattern. Azimuth/Elevation of the 3 jammers was 30/60(A) ,140/60(B), and 280/60(C). Fractional bandwidth of jammers was 0%(A) ,5% (B) , 30%(C). All jammers were 0 dB. JNR=40 dB.

Gain (dB)



(a)



(b)

Elevation

6x6 2D narrowband array
with omni-directional quiescent
pattern. Azimuth/Elevation of the 3
jammers was 30/60(A) ,140/60(B),
and 280/60(C). Fractional bandwidth
of jammers was 0%(A) ,5% (B) ,30%(C).
All jammers were 0 dB. 40 dB JNR.

Figure 4.34

Beampatterns (at f=1.0) at Jammer Azimuths for Optimal Planar Broadband and Narrowband Arrays.

(Example 4.15)

End of Example 4.15

4.3.3 Achieving Multiple Nulls in Broadband Arrays

Techniques which solve the Wiener-Hopf equation, or related equations for a broadband array, typically place a single null on each narrowband jammer and a sequence of nulls on each broadband jammer. The sequence of nulls is placed so as to null the broadband jammer over its entire bandwidth. A broadband array can provide independent control over spatial and frequency response, due to the presence of both spatially and temporally placed elements. Thus, the broadband array can place the sequence of nulls so as to achieve nulling over a wide frequency bandwidth while preserving a sharp null in azimuth.

We have developed techniques which automatically place multiple nulls on or near narrowband and broadband jammers using a broadband array. The techniques are similar to those presented in earlier sections of this chapter dealing with narrowband arrays. The typical operation for producing a multiple null in a broadband scenario is to linearly operate on each stacked snapshot so as to create one or more related synthetic snapshots, which together with the actual snapshot, emulate snapshots from multiple closely spaced emitters. Equivalently, the operation can be performed directly on the stacked covariance matrix. The stacked snapshot and stacked covariance matrix include data from both ***sensors and taps*** as opposed to the narrowband array which included data from only sensors.

Since a broadband array can provide independent control of spatial and frequency response, it is possible to achieve a number of different types of multiple nulls. Specifically, it is possible to achieve multiple nulls with respect to angle at one frequency, multiple nulls with respect to frequency at one angle, and multiple nulls with respect to both frequency and angle.

From a spectral point of view, multiple nulls in frequency extend the frequency band in which jamming signals are suppressed beyond the band actually occupied by the jamming signals. This feature is potentially useful in applications in which one wishes to avoid the nulling system problems associated with presence of the user signal. Specifically, as illustrated in Example 4.10 for a narrowband array, it is possible in some cases to use a broadband array to measure the jamming energy in a frequency band not occupied by the user signal and then to use multiple nulls to extend the bandwidth of the nulls. The utilization of a broadband array provides the opportunity to do this without sacrificing the spatial response of the array, i.e. without compromising the sharpness of the null with respect in angle.

From a spatial point of view, multiple nulls extend the angular sector over which jammer suppression is provided. This feature is potentially useful when there exists relative motion between the adaptive array and the jammers. In this case, nulls formed from collected data may no longer be effective due to a change in the relative position of one or more jammers. The use of multiple nulls in angle facilitates placing nulls not only at the angles appropriate to the collected data but also over a range of nearby angles. Thus the nulls can be automatically placed so as to anticipate any relative changes in angle due to platform motion.

Because of the similarity between the theory of multiple nulls in narrowband arrays and broadband arrays, we will only outline the procedures for producing the higher-order and clustered types of nulls in linear broadband arrays. The application to planar broadband arrays can be similarly derived. Elevation nulls in broadband arrays are not quite as useful as in narrowband arrays since the correspondence between frequency and elevation angle does not occur for broadband arrays (independent control over elevation angle response and frequency response can be achieved in a planar broadband array).

4.3.4 Producing Higher-Order Nulls (Broadband Arrays)

Consider a linear array of K sensors and L taps. We will assume that both the sensor spacing and tap spacing are uniform, although the theory can also be extended to non-uniformly spaced and sampled broadband arrays. The response of the array at the angle θ and frequency f is given by

$$G(\theta, f) = \sum_{l=0}^{L-1} \sum_{k=0}^{K-1} w_{kl} e^{j(2\pi f \frac{d}{c} \sin\theta)k} e^{-j(2\pi f T_s l)l} \tag{4.80}$$

where d denotes the spacing of the sensors in space and T_s denotes the spacing of the taps in time. The weight set $\{w_{kl}\}$ denotes the complex gain and phase which is applied to the l^{th} tap of the k^{th} sensor.

Provided that sufficient degrees of freedom are available, the broadband array will null a high JNR jammer arriving from angle θ_0 and frequency f_0 by adjusting the weights w_{kl} so that

$$G(\theta_0, f_0) = \sum_{l=0}^{L-1} \sum_{k=0}^{K-1} w_{kl} e^{j(2\pi f_0 \frac{d}{c} \sin\theta)k} e^{-j(2\pi f_0 T_s l)l} = 0 \tag{4.81}$$

The derivative of the response with respect to angle is given by

$$\frac{\partial G(\theta,f)}{\partial\theta}=\sum_{l=0}^{L-1}\sum_{k=0}^{K-1}w_{kl}(j2\pi f\frac{d}{c}\cos\theta k)e^{j(2\pi f\frac{d}{c}\sin\theta)k}e^{-j(2\pi fT_sl)l} \tag{4.82}$$

From equation (4.82) we see that a zero in the gain derivative with respect to angle can be achieved by synthesizing an additional stacked snapshot. This snapshot is derived from the true stacked snapshot by multiplying the $(kl)^{th}$ element of the broadband array by k. Thus a flat gain derivative at (θ_0, f_0) can be achieved by multiplying the data at all of the taps at the k^{th} sensor by the integer k. Note that if the jammer is comprised of multiple frequency components, the flat derivative in angle is achieved at each frequency. The multiplication by a ramp in k can be applied either to snapshot data or directly to the stacked covariance matrix with analogy to the narrowband discussion.

Similarly, we can compute the derivative of the gain with respect to frequency yielding

$$\frac{\partial G(\theta,f)}{\partial f}=\sum_{l=0}^{L-1}\sum_{k=0}^{K-1}w_{kl}(j2\pi\frac{d}{c}\sin\theta k-j2\pi T_sl)e^{j(2\pi f\frac{d}{c}\sin\theta)k}e^{-j(2\pi fT_sl)l} \tag{4.83}$$

In order to force the array to exhibit flat derivative with respect to frequency at (θ_0, f_0) , it is necessary to multiply the $(kl)^{th}$ element of each snapshot by $(j2\pi\frac{d}{c}\sin\theta_0k - j2\pi T_sl)$. Note that this quantity depends on the arrival angle θ_0. In some cases this can be estimated via coarse direction-of-arrival information and the proper correspondence between sensor and tap weighting can be maintained. Also note that sufficient conditions for

$$\frac{\partial G(\theta,f)}{\partial f}|_{\theta=\theta_0,f=f_0}=\sum_{l=0}^{L-1}\sum_{k=0}^{K-1}w_{kl}(j2\pi\frac{d}{c}\sin\theta_0k-j2\pi T_sl)e^{j(2\pi f_0\frac{d}{c}\sin\theta_0)k}e^{-j(2\pi f_0T_sl)l}=0 \tag{4.84}$$

are that

$$\sum_{l=0}^{L-1}\sum_{k=0}^{K-1}w_{kl}ke^{j(2\pi f_0\frac{d}{c}\sin\theta_0)k}e^{-j(2\pi f_0T_sl)l}=0 \tag{4.85}$$

and

$$\sum_{l=0}^{L-1}\sum_{k=0}^{K-1}w_{kl}le^{j(2\pi f_0\frac{d}{c}\sin\theta_0)k}e^{-j(2\pi f_0T_sl)l}=0 \tag{4.86}$$

Thus, a flat derivative in frequency can be achieved by synthesizing *two* additional snapshots for each actual snapshot. The first additional snapshot is related to the true snapshot by a ramp in k for all tap data. The second snapshot is related to the true snapshot by a ramp in l for all sensor data. This method consumes an extra degree of freedom but does not require direction-of-arrival information. Note that the latter procedure for producing flat frequency derivative also forces flat angle derivative.

The following example illustrates the utilization of multiple nulls in a linear broadband array.

Example 4.16: Use of Higher-Order Nulls in Linear Broadband Arrays

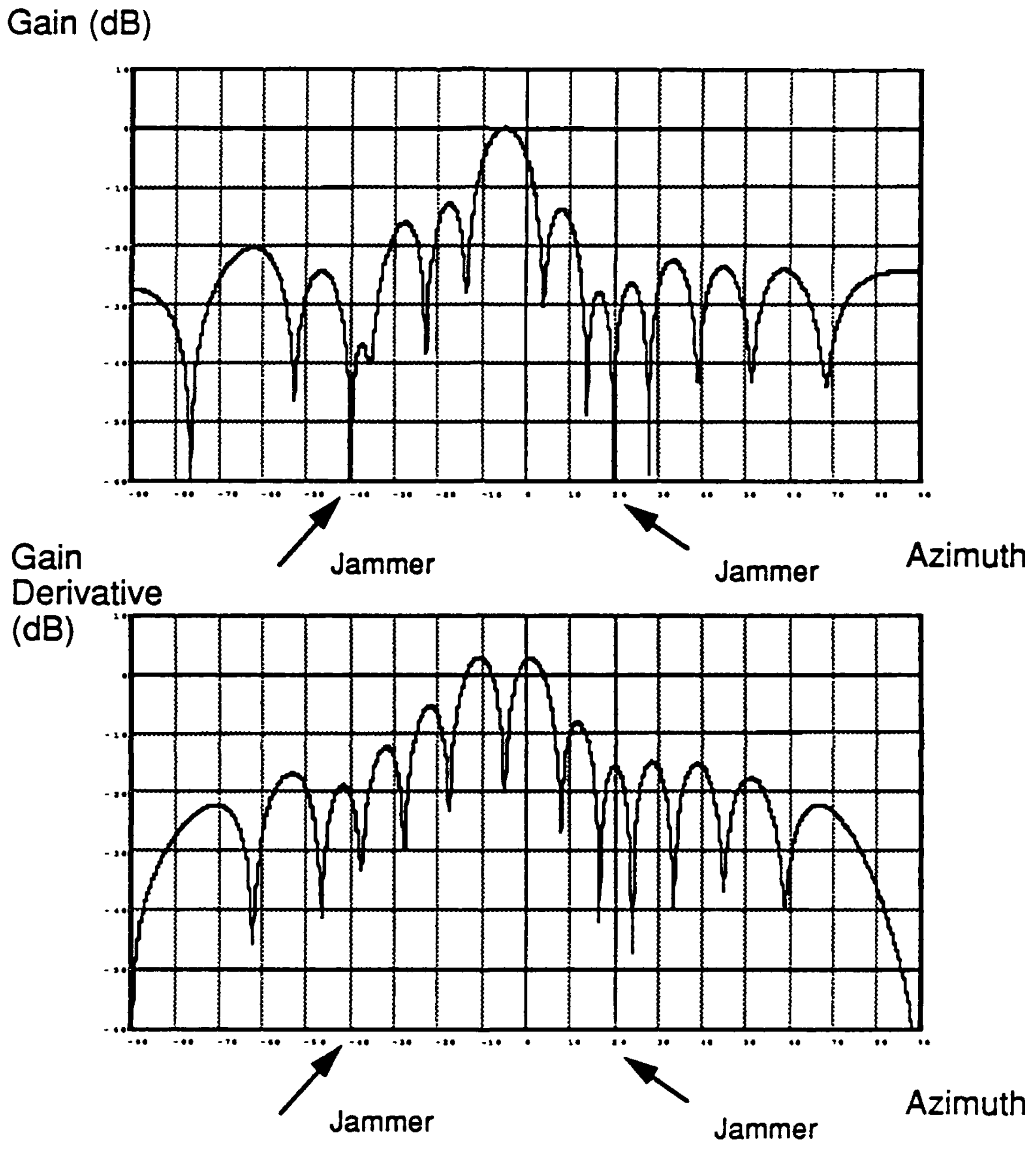
Consider a linear broadband array with 16 elements and 8 taps per element. The array was designed to steer across a 25 % fractional bandwidth (from $f = 0.75 - 1.0$) in the azimuth direction of $-5°$. The array adapted in the presence of two 40 dB JNR broadband jammers at azimuths of $-40°$ and $20°$. Conventional processing (i.e. the Wiener equation) was used to determine the optimal weight set for this broadband array.

The corresponding beampattern, as a function of azimuth, for an arbitrarily chosen frequency of $f = 0.8$, within the fractional bandwidth of the jammers, is depicted in the top curve of Figure 4.35. Note that the array exhibits unity gain in the constraint direction and places nulls in the directions of the two jammers. The bottom curve in Figure 4.35 depicts a scaled version of the derivative of the gain pattern with respect to azimuth. Note that the array places nulls on the jammers but does not force the derivative of the pattern to be zero at the jammer locations.

The weights were then re-computed using multiple-null processing (second-order nulls) for the broadband array. The processing consisted of multiplying the sensor data by a ramp in x, i.e. all taps at sensor 0 were multiplied by 0, all taps at sensor 1 were multiplied by α, all taps at sensor 2 were multiplied by 2α, etc., and was accomplished by operating directly on the steady-state stacked covariance matrix. The constant α was chosen so as to make the ramp have unit norm. The corresponding beampattern and derivative were computed for the same frequency as above ($f = 0.85$) and are plotted in Figure 4.36.

Note that the nulls placed on the jammers are now wider in angle and the gain derivative with respect to azimuth is also zero at the two jammer locations. Similar results were noted for other frequencies within the jammer frequency band. Figure 4.37 depicts the magnitude of the frequency response in the steered direction and the directions of the two jammers. The gain of the array is flat (0 dB) over the constrained band and nearly zerc over the jammer bandwidths. Clearly, higher-order processing for the broadband array forced both nulls and flat azimuth derivative in the directions of the jammers while maintaining wide nulls in frequency.

End of Example 4.16

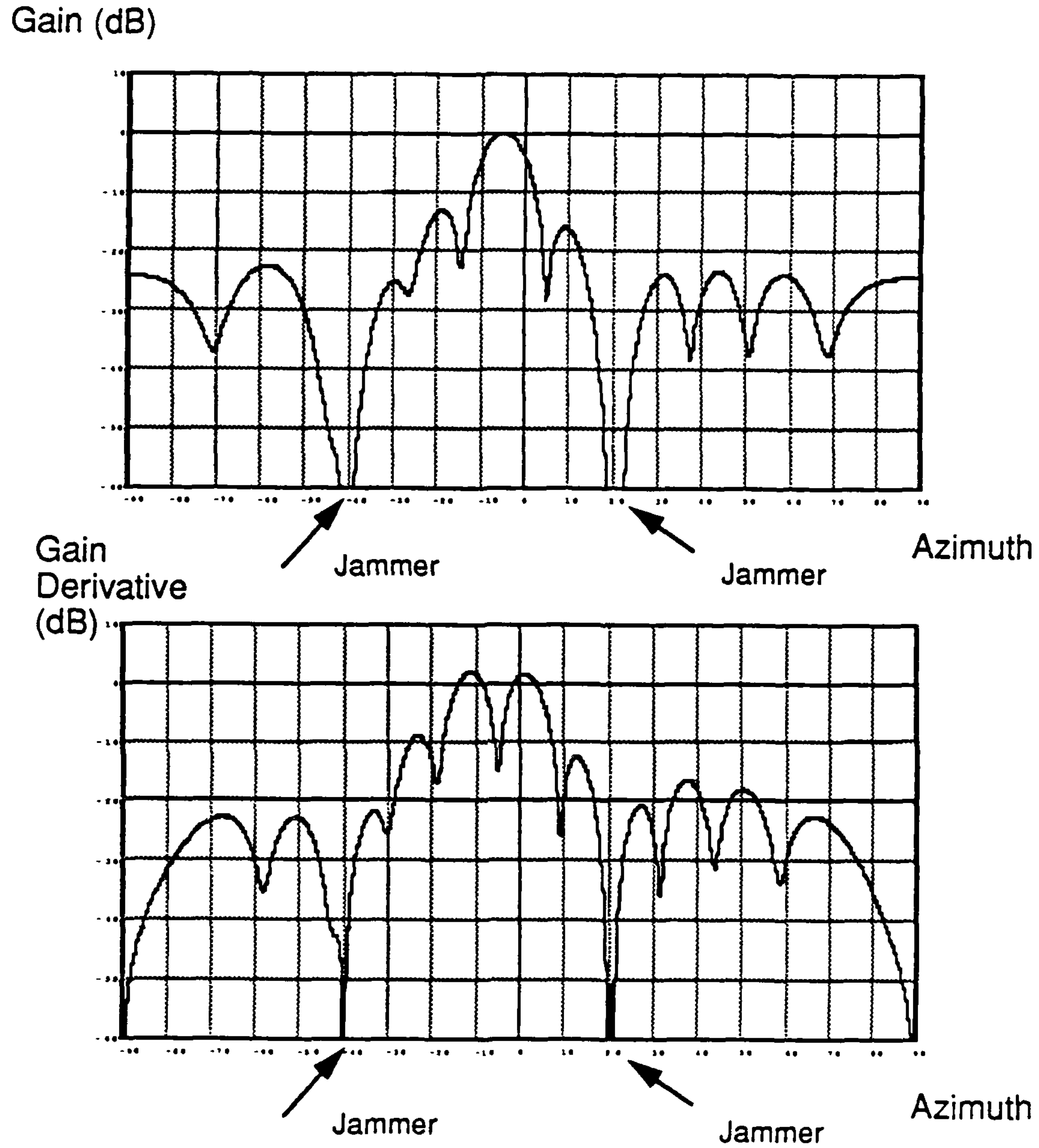


16 Element linear broadband array with 8 taps per element. 4 constraints were used to steer the array toward a 25% fractional bandwidth user at -5 degrees. 0dB jammers with 25% fractional bandwidth at 20 degrees and -40 degrees. JNR=40 dB.

Figure 4.35

Beampattern (Gain) and Derivative for Optimal Linear Broadband Array (at f =0.85). Conventional processing places first order nulls only.

(Example 4.16)



16 Element linear broadband array with 8 taps per element. 4 constraints were used to steer the array toward a 25% fractional bandwidth user at -5 degrees. 0dB jammers with 25% fractional bandwidth at 20 degrees and -40 degrees. JNR=40 dB.

Figure 4.36

Beampattern (Gain) and Derivative for Optimal Linear Broadband Array (at f =0.85). Higher-order null processing forced zero gain and zero gain derivative in jammer directions.

(Example 4.16)

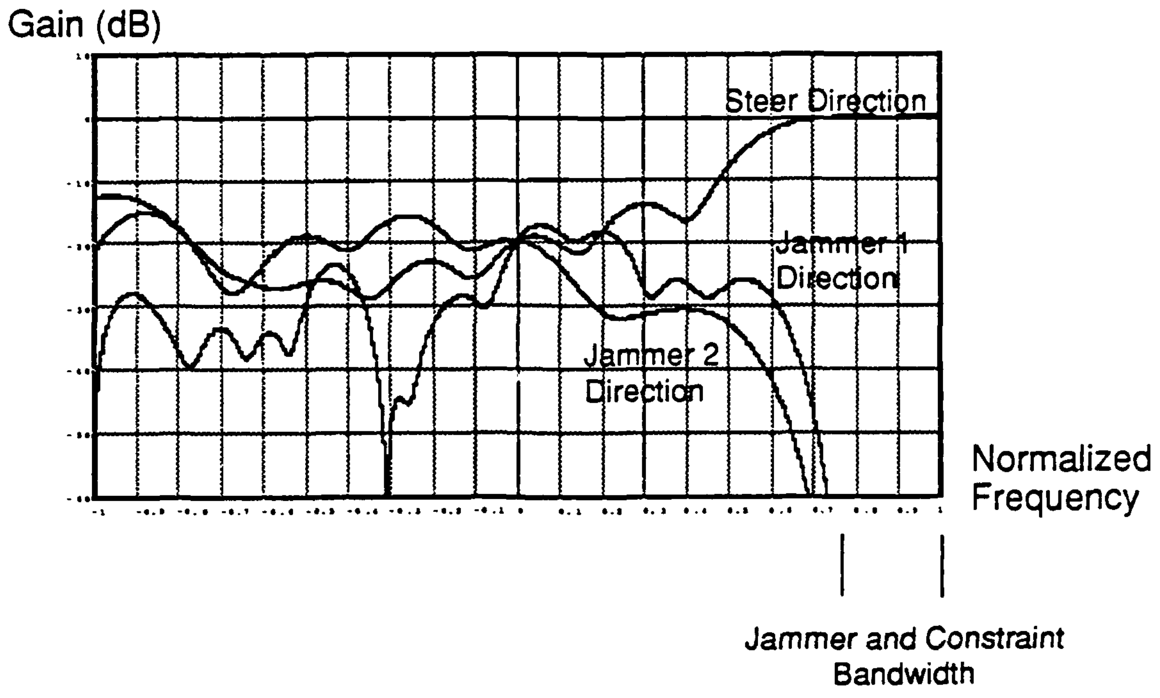


Figure 4.37

Frequency Response (Magnitude) in Jammer Look Directions and Steer Direction for an Optimal Linear Broadband Array (at f =0.85). Higher-order null processing preserved wideband nulls.

(Example 4.16)

16 Element linear broadband array
with 8 taps per element. 4 constraints
were used to steer the array
over a 25% fractional bandwidth
at -5 degrees. 0dB jammers
with 25% fractional bandwidth at
20 degrees and -40 degrees. JNR=40 dB .

4.3.5 Producing Clustered Nulls in Broadband Arrays

Clustered nulls can be achieved in broadband arrays through the use of synthetic stacked snapshots generated by appropriately modulating each actual stacked snapshot. The form of the modulation determines whether the nulls are clustered along angle for a fixed frequency, along frequency for a fixed angle, or along both frequency and angle.

To derive the appropriate modulation factor, we represent equation (4.80) as

$$G(\theta, f) = \sum_{l=0}^{L-1} \sum_{k=0}^{K-1} w_{kl} e^{jw_x k} e^{jw_T l} \tag{4.87}$$

where

$$w_x = 2\pi f_d \sin\theta / c \tag{4.88}$$

and

$$w_T = -2\pi f T_s \tag{4.89}$$

Note that if each snapshot is modulated so that the $(kl)^{th}$ element is multiplied by $e^{j\delta_x k} e^{\delta_T l}$ then a given component of the synthetic snapshot will appear as having arrived from some different angle and frequency. In particular, for modulation factors of δ_x and δ_T the new arrival angle θ_N and frequency f_N will be given by

$$\theta_N = \sin^{-1}[\frac{T_S}{d}(\frac{w_x + \delta_x}{w_T + \delta_T})] \tag{4.90}$$

and

$$f_N = -\frac{w_T + \delta_T}{2\pi T_s} \tag{4.91}$$

If $\delta_T = 0$, the nulls will be clustered along angle only and the spacing of the nulls in the cluster can be predicted using equation (4.90). Note that the spacing in angle between nulls in a cluster is dependent on frequency. Thus, the cluster will spread over a range of angles for broadband jammers.

Clustering nulls in frequency only is more difficult to achieve and requires coarse directional information. A series of clustered nulls can be achieved by synthesizing a series of additional snapshots. The clustered null processing can also be performed by operating directly on the stacked covariance matrix with analogy to the narrowband discussion presented earlier in this chapter.

The following example illustrates the use of clustered nulls in linear broadband arrays.

Example 4.17: Use of Clustered Nulls in Linear Broadband Arrays

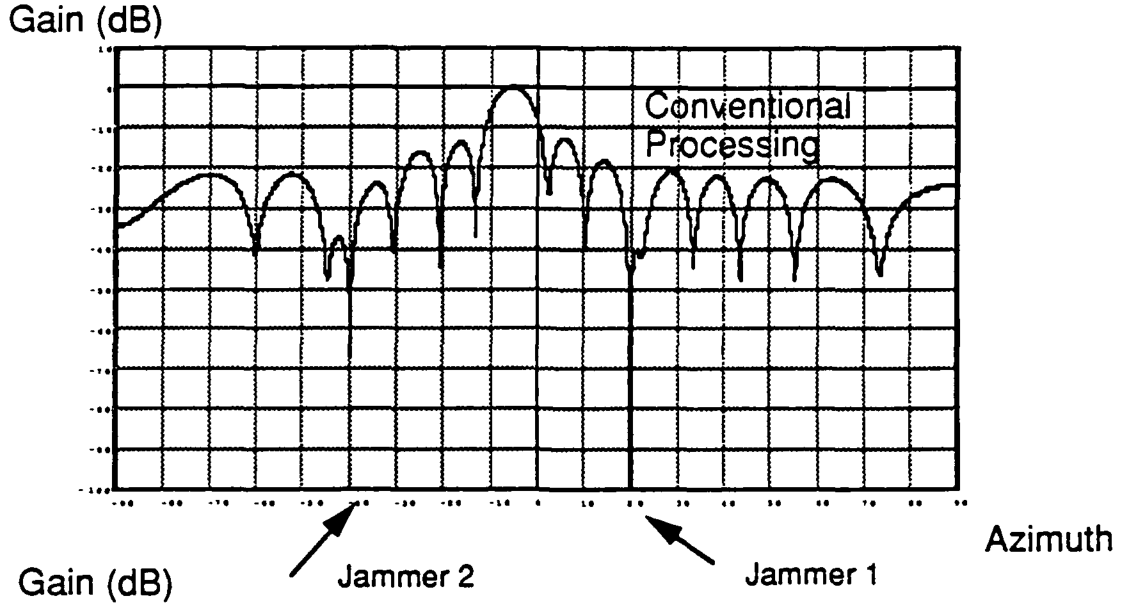
Consider the same broadband scenario as in Example 4.16 consisting of two broadband jammers at $-40°$ and $20°$ and 16 element , 8 tap per element broadband array. The weights for this array were determined using the conventional processing and the beampattern at an arbitrarily chosen frequency of $f = 0.95$ within the jammer band was computed. The magnitude of this beampattern is depicted in the top curve of Figure 4.38.

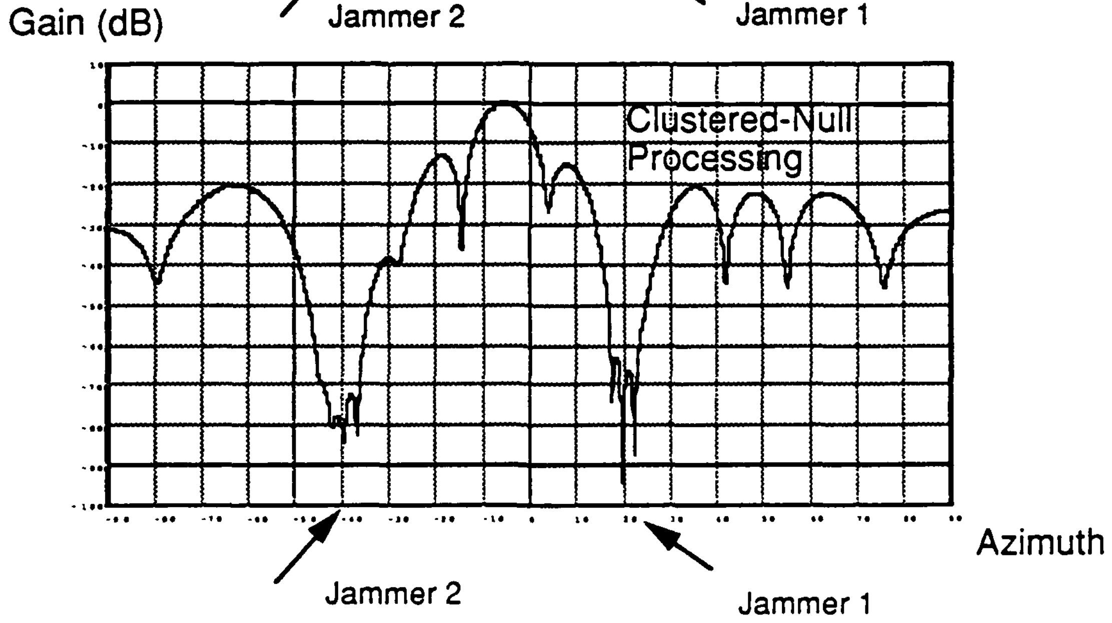
The weights were then recomputed using clustered null processing. Two additional nulls were achieved by modulating the array data along x. The first extra null in the cluster was achieved by choosing the modulation factor of $+\delta x$, i.e. all taps at sensor 0 were shifted in phase by 0, all taps at sensor 1 were shifted in phase by δx, all taps at sensor 2 were shifted in phase by $2\delta x$. The second extra null in the cluster was achieved by choosing the modulation factor of $-\delta x$. The modulation value δx was chosen as .1.

The beampattern for the weight set with clustered-null processing in plotted in the bottom curve of Figure 4.38. The additional two nulls in the vicinity of each of the jammers are evident. The width of the null cluster is dependent on frequency - different beampatterns at frequencies within the jammer band exhibit different angular spreads in the cluster.

The overall effect of this form of clustered-null processing is to broaden the nulls in angle around each broadband jammer. This effect can be seen more clearly on Figure 4.39. The top curve depicts the broadband beampattern for the optimal broadband array with conventional processing. The bottom curve clearly depicts the broadening of the nulls due to clustered-null processing.

End of Example 4.17



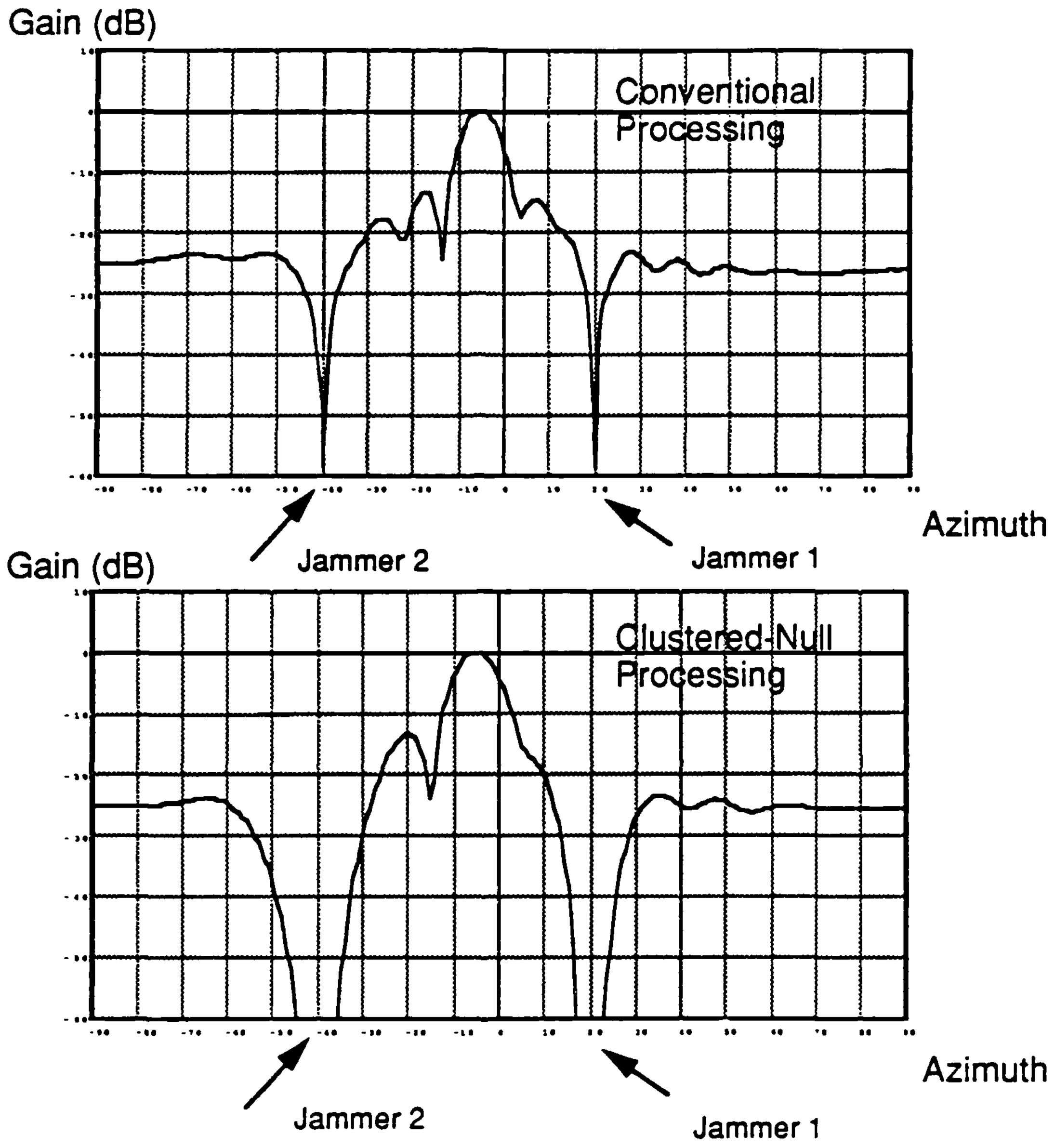


16 Element linear broadband array with 8 taps per element. 4 constraints were used to steer the array over a 25% fractional bandwidth at -5 degrees. 0dB jammers with 25% fractional bandwidth at 20 degrees and -40 degrees. JNR=40 dB .

Figure 4.38

Beampatterns (at f=0.9) for Optimal Linear Broadband Array with Conventional Processing (top curve) and Clustered-Null Processing (bottom curve).

(Example 4.17)



16 Element linear broadband array with 8 taps per element. 4 constraints were used to steer the array over a 25% fractional bandwidth at -5 degrees. 0dB jammers with 25% fractional bandwidth at 20 degrees and -40 degrees. JNR=40 dB.

Figure 4.39

Broadband Beampatterns (over f=0.75-1.0) for Optimal Linear Broadband Array with Conventional Processing (top curve) and Clustered-Null Processing Processing (bottom curve).

(Example 4.17)

Bibliography

[1] Robert A. Monzingo and Thomas W. Miller, *Introduction to Adaptive Arrays*, John Wiley and Sons, 1980.

[2] J. E. Hudson, *Adaptive Array Principles*, Peter Peregrinus Ltd., New York, 1981.

[3] H. Steyskal, "Wide-band nulling performance versus number of pattern constraints for an array antenna", *IEEE Trans. Antennas Propagat.*, vol. AP-31, pp.159-163, Jan. 1983.

[4] K. Takao and K. Komiyama, "An adaptive antenna for rejection of wideband interference", *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-16, pp.452-459, July 1980.

[5] R. Riegler and R. Compton Jr., "An adaptive array for interference rejection", *Proc. IEEE*, vol. 61(6), pp. 748-758, June 1973.

[6] K. Buckley and L. Griffiths, "An adaptive sidelobe cancellor with derivative constraints", *IEEE Trans. Antennas Propagat.*, vol.AP-34, pp.311-319, Mar. 1986.

[7] K. Buckley "Spatial/spectral filtering with linearly constrained minimum variance beamformers", *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-35, Mar. 1987.

[8] M. Er and A. Cantoni, "Derivative constraints for broad-band element space antenna array processors", *IEEE Trans. Acoust., Speech, Signal Processing*, vol.ASSP-31, pp.1378-1393, Dec. 1983.

[9] W.F. Gabriel, "Using spectral estimation techniques in adaptive processing antenna systems", *IEEE Trans. Antennas Propagat.*, Mar. 1986.

[10] "Advanced Nulling Techniques: Phase I Final Report", Atlantic Aerospace Electronics Corporation (Formerly Pollard Road Inc.), AF Contract Number F04701-85-C-0078, Mar. 1986.

Chapter 5

Effects of Pointing and Hardware Errors

Contents

List of Figures

5.1 Introduction

The purpose of this chapter is to assess the performance of the new class of algorithms of Section 3.3 under "non-ideal" operating conditions which include pointing errors, channel hardware errors.

The main conclusions are as follows:

1. The presence of a user signal in the data used to calculate nulling weights greatly ***magnifies*** the adverse impacts of pointing errors and hardware errors. The effect of such errors is to change the apparent angle of arrival of the user signal. The nulling processor then mistakes the user signal for a mainbeam jammer and attempts to null the user along with the jammers.

2. The steady-state interference-suppression performance of the new class of algorithms described in Chapter 3 is much less sensitive to both pointing errors and channel hardware errors than many conventional algorithms in the user-present case.

3. A variety of "fixes" have been proposed in the literature to reduce the sensitivity of conventional null algorithms to the foregoing types of errors[1][2][3][4][5][6]. The performance of the new class of algorithms with no fixes is comparable to that of some conventional algorithms with fixes. When corresponding fixes are added to the new class of algorithms, they typically exhibit improved performance (or reduced sensitivity).

4. The sensitivity of the new class of algorithms to the foregoing types of errors ***under transient conditions*** is far superior to that of many conventional nulling algorithms with fixes.

The chapter is organized as follows. Section 5.2 reviews the pointing error problem for the user-present case, and compares the performance of the new algorithms to a number of conventional approaches. Section 5.3 reviews the channel hardware error problem for the user present case and shows that the effects are similar to those of pointing errors; comparison of the new algorithms with conventional algorithms are presented. The conventional approaches addressed in Sections 5.2 and 5.3 include utilization of linear constraints, norm constraints, and thinned beam methods.

5.2 Techniques for Reducing Impact of Pointing Errors

5.2.1 Introduction

The purpose of this section is to describe the pointing error problem in adaptive arrays and to review both classical and new methods for reducing the impact of pointing errors.

5.2.2 Impact of Pointing Errors on Performance

If the user signal is present in the snapshots used to calculate the sample covariance matrix $\hat{R}_{xx}$, and the actual user direction is only in slight error relative to the direction assumed for the vector $\vec{u}_0$, then the ***resultant weight vector will attempt to null the user.*** In effect the SMI processor mistakes the user for a mainbeam jammer. This effect was described by Cox[8], who used the term *mismatch* to designate the discrepancy between the actual and assumed user directions. Cox showed that even small values of mismatch can produce dramatic degradation for a signal SNR greater than 0 dB. The following example illustrates this effect.

Example 5.1: Effect of User Mismatch for the Wiener Weight Vector

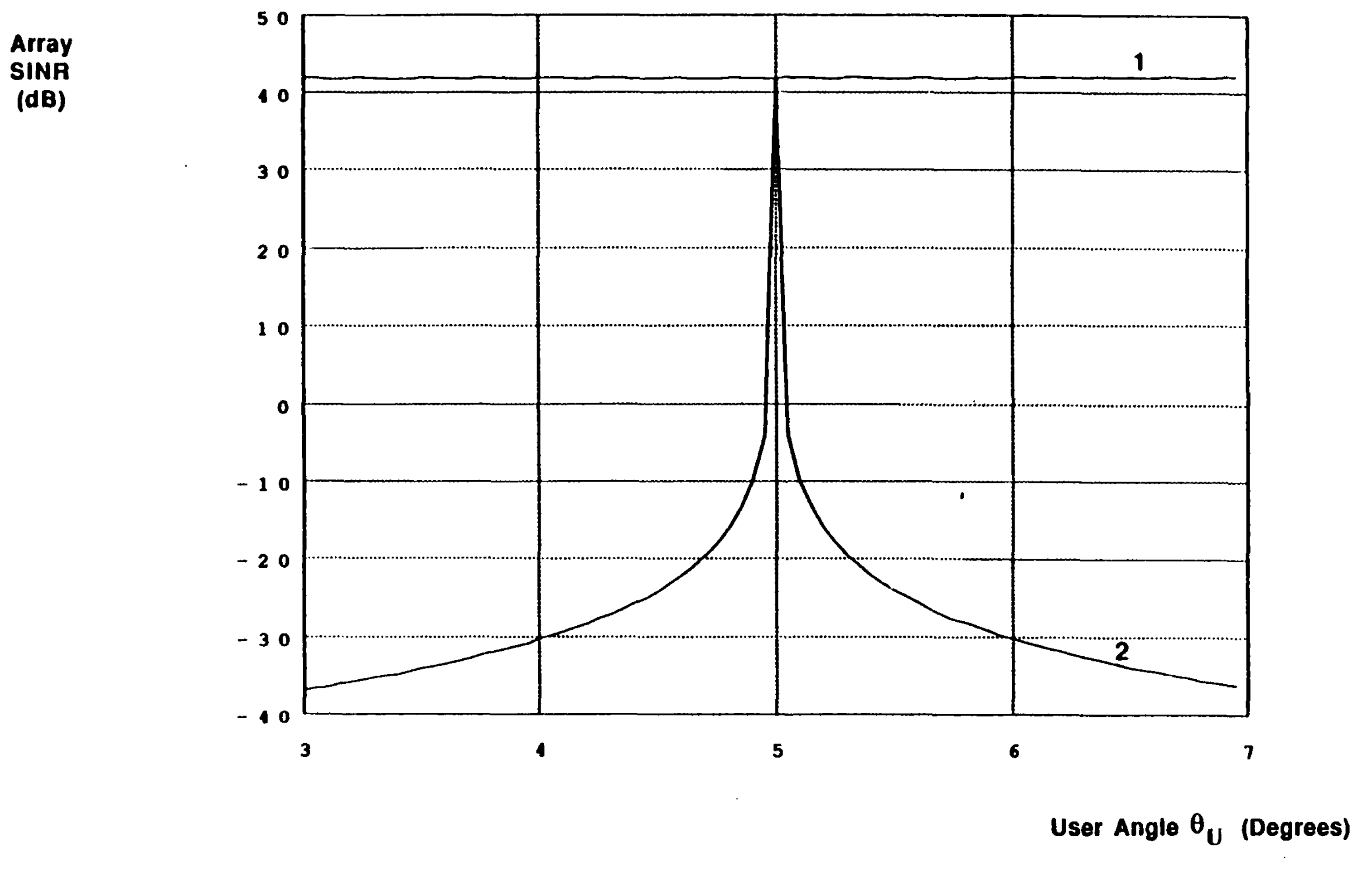
Consider again the scenario of Example 3.1 modified so that the user signal transmits from a parametric angle θ_U rather than at the fixed angle of 5°. The weight vector that solves the *exact* Wiener-Hopf equation was used to calculate the array SINR as the angle θ_U of the user was varied.

As a baseline, we assumed that the 30 dB user signal arrives from the angle θ_U and that $\vec{u}_0$ was a unit-arrival vector for the user at θ_U. Thus, for the baseline case, the steering vector $\vec{u}_0$ was perfectly matched to the user direction. The corresponding weight vector was determined for a small range of values of θ_U. The results are depicted by Curve 1 in Figure 5.1. Note that a high, nearly constant, array SINR is achieved over the entire range of θ_U.

Next we repeated the procedure assuming that the user signal again arrives from an angle θ_U, but that a fixed steering vector $\vec{u}_0$ is used corresponding to the (erroneous) direction of 5°. The resultant SINR as a function of θ_U is plotted as Curve 2 in Figure 5.1. Note the extreme rate at which the output SINR falls as the mismatch between the fixed steering vector $\vec{u}_0$ and the actual user direction increases.

Figure 5.2 depicts the antenna pattern for Wiener weights when the steering vector is at 5° and the actual user is present at an angle 5.1°. Clearly, the pattern provides unity gain

Figure 5.1: Impact of User Signal Presence on Pointing Errors



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector (Weights Computed Using Steady-State Weiner Solution)

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data for All 3 Cases
30 dB JNR Jammers at 15, 50 Degrees

Figure 5.2a: Expanded Antenna Pattern for Weight Vector Calculated for 0.1° (= 5.1° - 5°) Mismatch (Example 5.1)

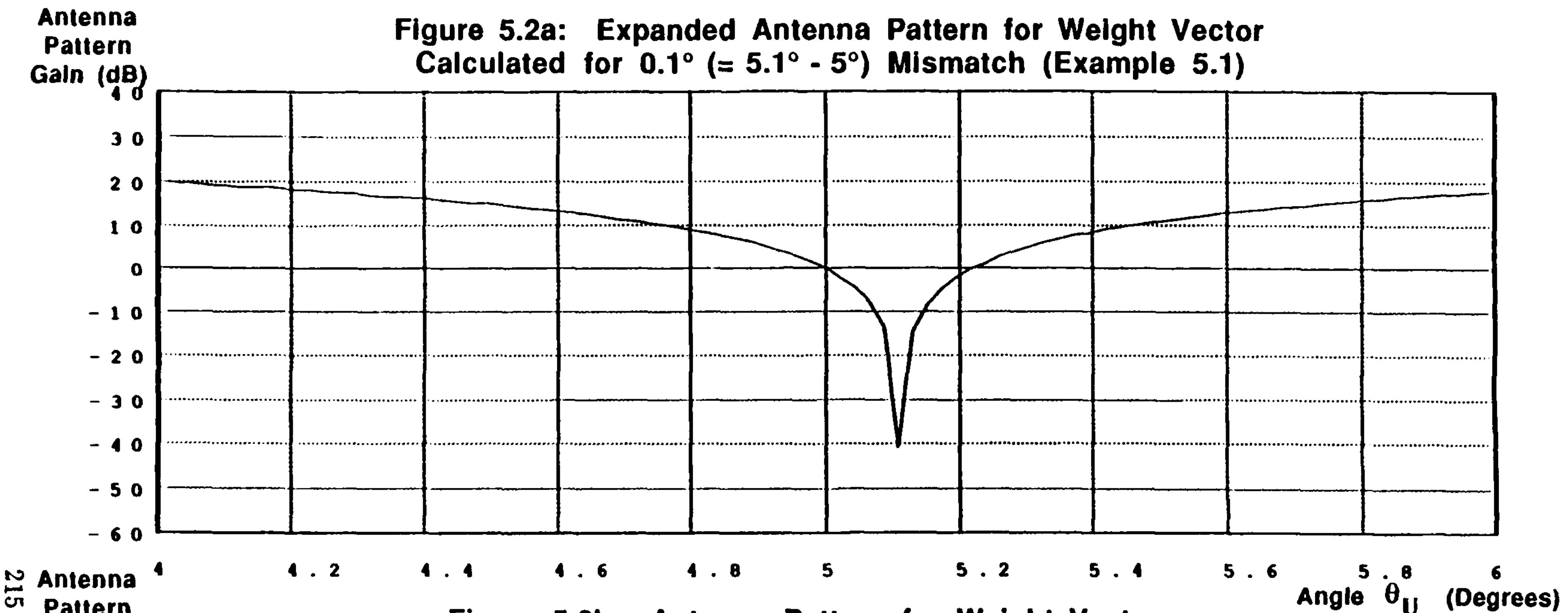
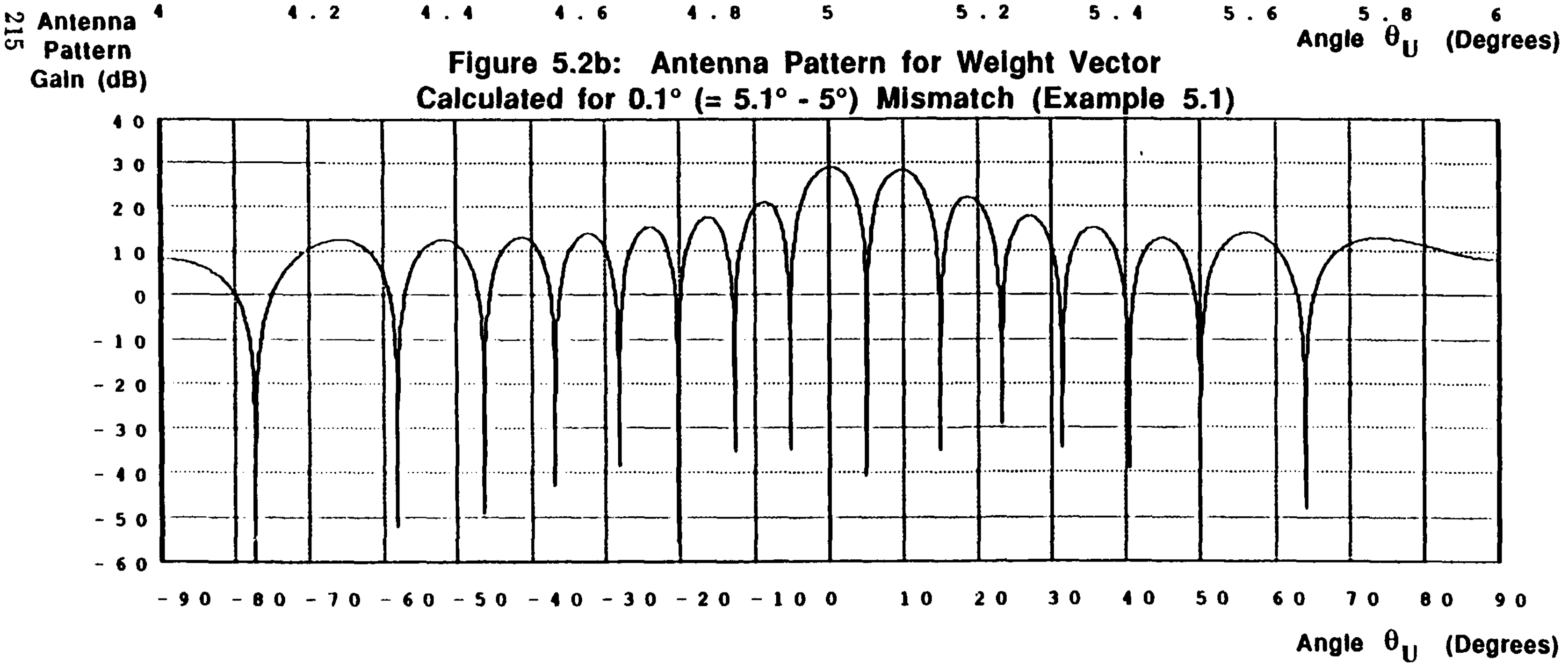


Figure 5.2b: Antenna Pattern for Weight Vector Calculated for 0.1° (= 5.1° - 5°) Mismatch (Example 5.1)



in the assumed direction of the user (i.e., 5°) while it nulls the user at the actual location of 5.1°. The very large peaks in the pattern reflect large weights. The large weights in turn cause the array to output a high level of thermal noise power, which accounts for the unfavorable SINR in Figure 5.1.

End of Example 5.1

In general, the user signal acts to ***magnify*** the impact of pointing errors. The stronger the user signal, the greater the sensitivity to pointing errors.

Example 5.2: Performance as a Function of User Level

To illustrate the importance of the user signal level, the experiment of Example 5.1 was performed for a range of user SNR values. The array gain after adaptation was determined for each user signal level as a function of the user arrival angle θ_U. The results are presented in Figure 5.3. The maximum array gain of $10\log_{10} K = 12$ dB was achieved when there was no mismatch between the user arrival angle θ_U and the steering vector. Array gain is defined as the ratio of the array output SINR to the individual sensor SNR. For large user signal SNR the array gain falls off rapidly as the mismatch angle departs from zero. Clearly, the sensitivity of array gain to mismatch drops dramatically as the user level approaches zero.

End of Example 5.2

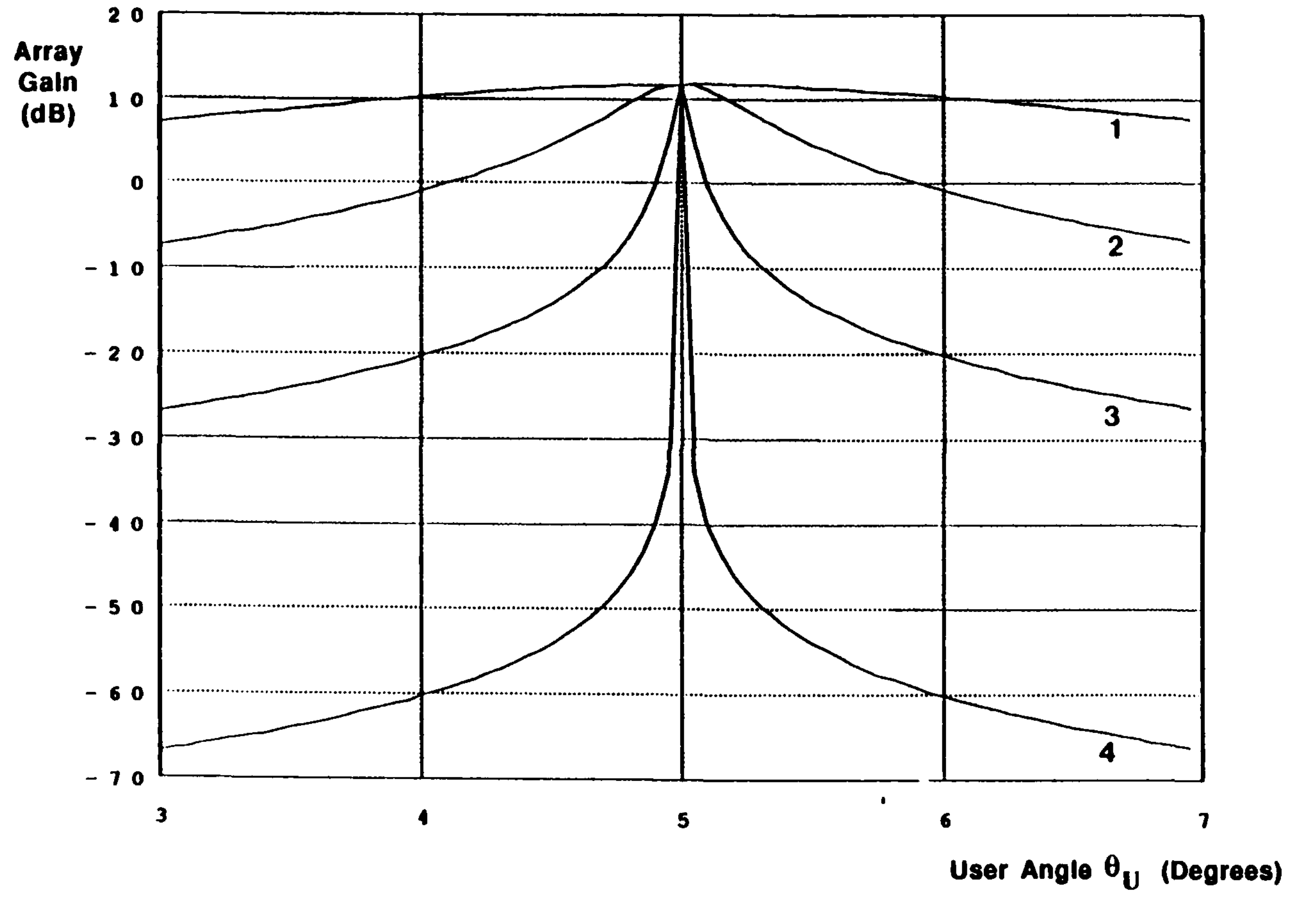
5.2.3 Use of Linear Constraints to Reduce Impact of User Signal on Mismatch

A classical technique for reducing the likelihood that the weight vector accidently nulls the user given mismatch is to impose either derivative constraints or multiple point constraints on the array antenna pattern[2][7]. The objective of the constraints is to keep the antenna pattern nearly constant in the vicinity of the assumed user direction [1][2][7]. In the case of derivative constraints it is required that the first M angular derivatives of the antenna pattern equal zero in the assumed direction of the user. In the case of multiple point constraints, it is required that the pattern approximate its maximum value at M closely-spaced points in the vicinity of the assumed user direction. The following examples illustrate the use of derivative constraints for the steady state ($N = \infty$) weight vector.

Example 5.3: Illustration of Derivative Constraints

The weight vector for the scenario of Example 5.1 was recalculated subject to constraints

Figure 5.3: Dependence of Pointing Error Sensitivity on User Signal Level (Example 5.2)



CURVES:

1. User Signal Absent
2. SNR = -10 dB
3. SNR = 0 dB
4. SNR = 30 dB

16-Element Linear Array
Variable SNR User Signal Present at Angle θ_U
30 dB JNR Jammers at 15, 50 Degrees

as follows.

> Case 1: The array antenna pattern has unity gain in the assumed direction $\theta = 5^\circ$ of the user, and also has a ***zero first derivative in this direction.***
>
> Case 2: The array antenna pattern has unity gain in the assumed direction of the user, and also has ***zero first and second derivatives in this direction.***

The SINR results are presented in Figure 5.4 for a range of values for the actual user angle θ_U. The curves of Figure 5.1 are replotted as Curves 1 and 2 for reference. Curve 3 depicts the SINR vs. θ_U behavior for Case 1; Curve 4 depicts the behavior for Case 2. Clearly, constraining the first derivative to zero at $\theta = 5^\circ$ reduces sensitivity to pointing errors. Additionally, requiring that the second derivative equals zero further reduces sensitivity.

End of Example 5.3

5.2.4 Use of Norm Bound to Reduce Impact of Pointing Errors

In general, derivative and point constraints are most effective at protecting an angular sector against inadvertent nulling of the user at moderate values of user SNR. As the SNR is increased, the protected angular region is reduced [2].

An alternative approach to containing the impact of pointing error is to upper bound the norm of the weight vector [3] [4]. This amounts to imposing an inequality constraint of the form

$$\vec{w}^H \vec{w} \leq b \tag{5.1}$$

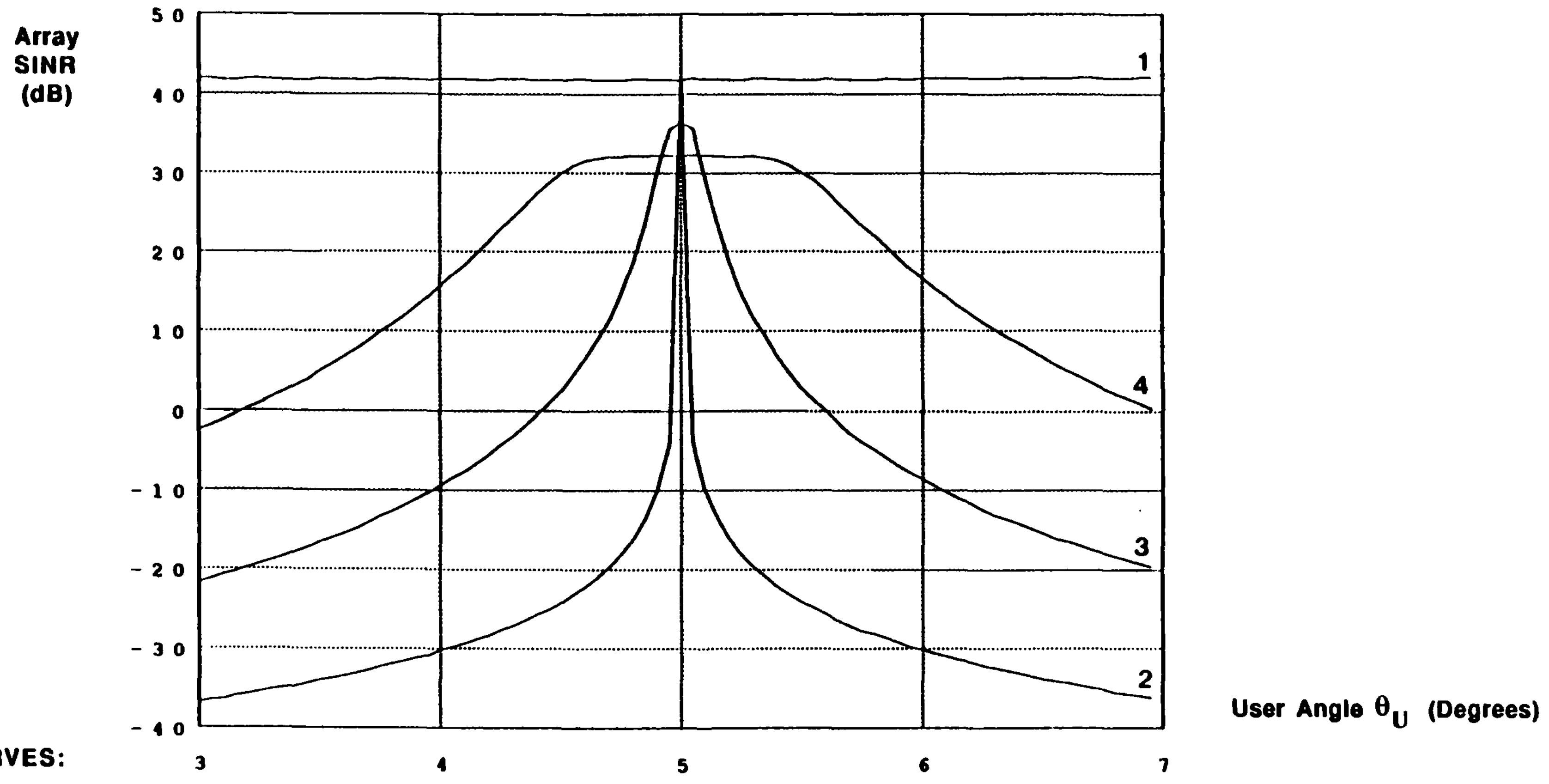
on the power minimization problem where b is a suitably selected positive constant. The constraint (5.1) effectively bounds the thermal noise power output by the array.

In general, the constrained minimization problem is solved in the following manner. First the constraint (5.1) is ignored and the Wiener weight vector $\vec{w}$ is calculated in the normal manner. If the norm of the resultant vector satisfies (5.1), then the vector is used in the adaptive processor. Otherwise, an equality constraint of the form

$$\vec{w}^H \vec{w} = b \tag{5.2}$$

is imposed. Power minimization subject to (5.2) and a linear gain constraint leads to a

Figure 5.4: Use of Derivative Constraints to Reduce Impact of Pointing Error (Example 5.3)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector (Constraints: Antenna Patter Has Unity Gain at $\theta_U = 5°$)
3. SINR as a Function of User Position for Fixed 5 Degree Steering Vector (Constraints: Antenna Pattern Has Unity Gain and Zero First Derivative at $\theta_U = 5°$).
4. SINR as a Function of User Position for Fixed 5 Degree Steering Vector (Constraints: Antenna Pattern Has Unity Gain and Zero First and Second Derivatives at $\theta_U = 5°$)

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U
30 dB JNR Jammers at 15, 50 Degrees

modified Wiener-Hopf equation of the form

$$(R_{xx} + \mu_0 I)\vec{w} = \mu\vec{u}_0 \tag{5.3}$$

Equation (5.3) is solved for $\vec{w}$, and the Lagrange multipliers μ_0 and μ are selected to satisfy the constraints. For the procedure to be effective in reducing the impact of pointing errors in high SNR user scenarios, the parameter b must be selected sufficiently small that the latter solution method is required.

It is evident from (5.3) that the effect of the constraint (5.2) is to add a *pseudo-noise* covariance matrix $\mu_0 I$ to the covariance matrix R_{xx}. Thus, the constraint (5.2) has the effect of increasing the noise floor by injecting a level of synthetic thermal noise along the covariance matrix diagonal. Increasing the thermal noise, in turn, has the effect of increasing the output power of the array when the array weights are large. Thus, the (inadvertent) suppression of the user signal is reduced by imposing the additional penalty function associated with large weights.

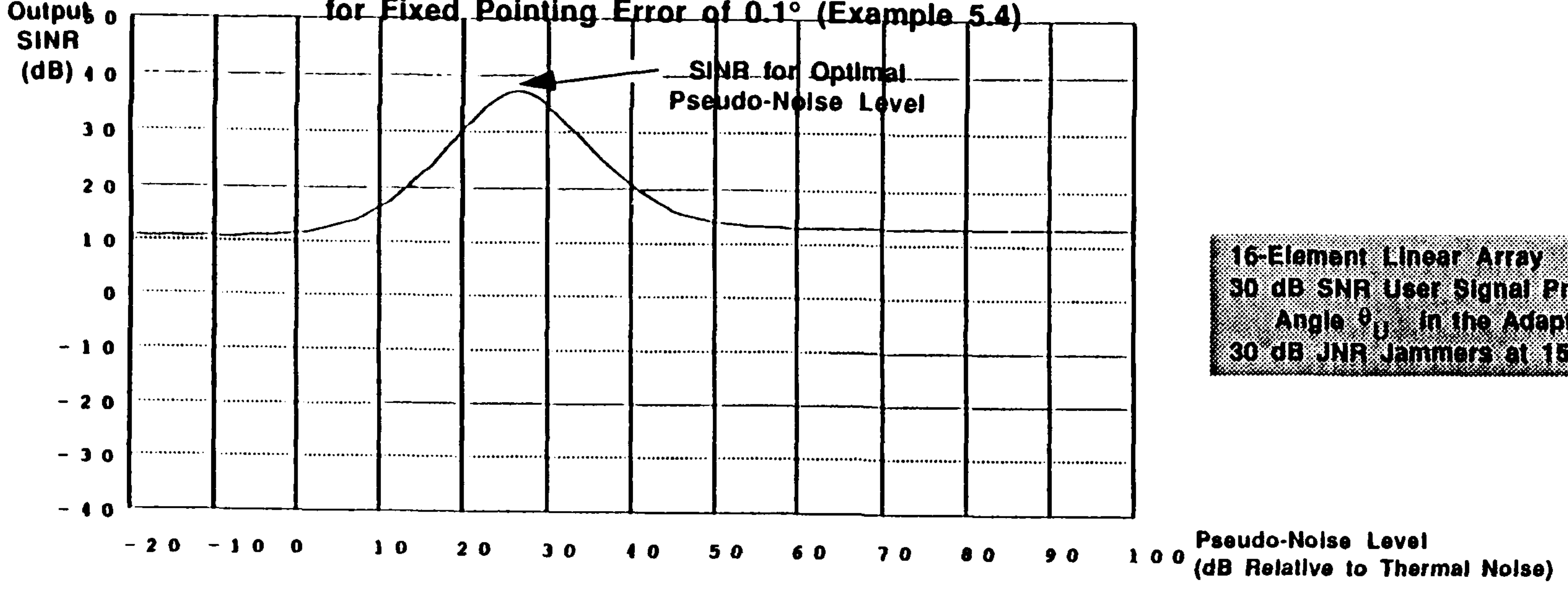
The following example illustrates the use of pseudo-noise and norm bounds.

Example 5.4: Illustration of Norm Bounds

The weight vector for the scenario of Example 5.1 was computed assuming a mismatch error of 0.1°, i.e. $\vec{u}_0$ was selected to steer at the assumed user direction of 5° while the actual user arrival angle was 5.1°. Figure 5.5a depicts the resultant array SINR as a function of the level of pseudo-noise added to the steady-state covariance matrix used to compute the associated weight set. The curve shows that the SINR increases as the level of pseudo-noise is increased, up to some optimal value of pseudo-noise, and then decreases to settle out at a final asymptotic value. The asymptotic SINR can be predicted a priori since the weight set is essentially unadapted from its quiescent value when the level of pseudo-noise is large.

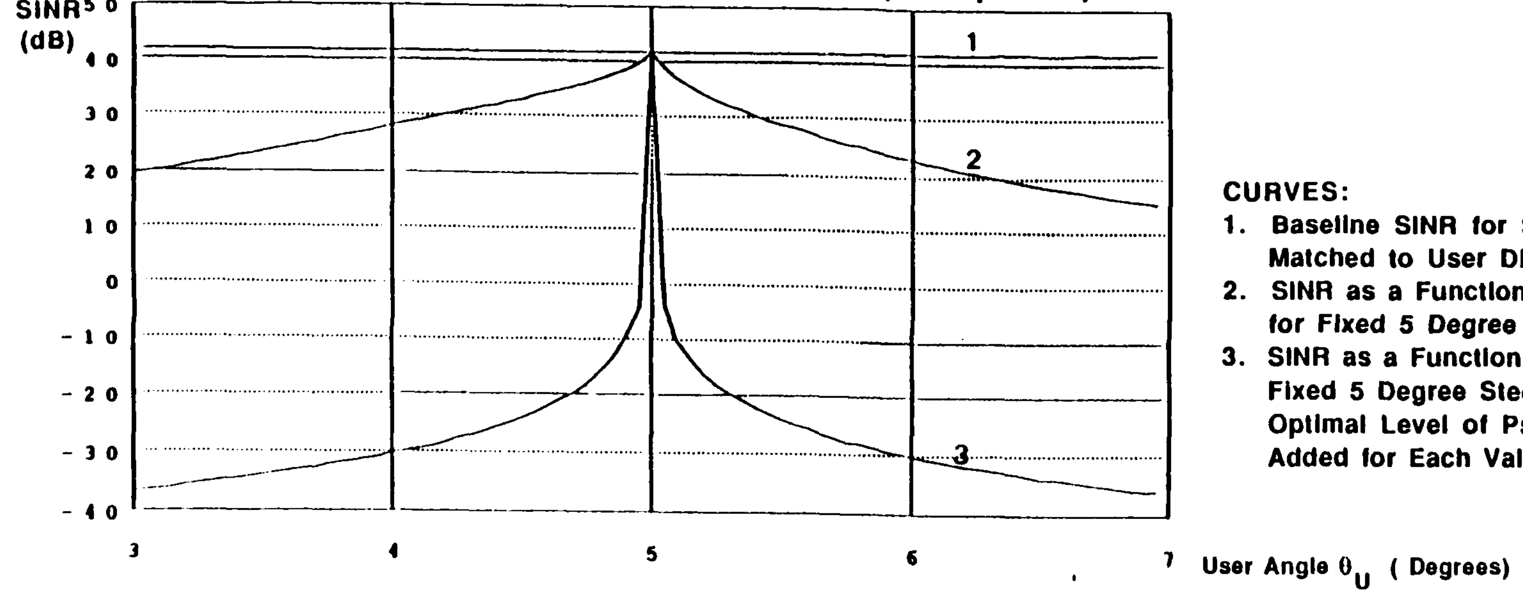
Figure 5.5b depicts the result of adding the *optimal level* of pseudo-noise to the covariance matrix as a function of the actual user direction θ_U. The scenario is again that of Example 5.1. Curves 1 and 2 in Figure 5.5b are provided as reference and are identical to Curves 1 and 2 in Figure 5.1. Curve 3 in Figure 5.5b depicts the SINR as a function of user arrival angle θ_U when the *optimal level of pseudo-noise* is added for each user position. The optimal level was determined iteratively for each value of θ_U by increasing the pseudo-noise level until the array SINR peaked. Alternate algorithms for determining this level have been discussed by Hudson[1]. Curve 3 in Figure 5.5b shows that the array SINR can be made to exhibit relatively robust performance to user pointing errors provided that the optimal value of pseudo-noise is utilized.

Figure 5.5a: Illustration of the Effect of Adding Pseudo-Noise for Fixed Pointing Error of 0.1° (Example 5.4)



16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 60 Degrees

Figure 5.5b: Improvement in Performance Due to Addition of Optimum Pseudo-Noise Level (Example 5.4)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector
3. SINR as a Function of User Position for Fixed 5 Degree Steering Vector. An Optimal Level of Pseudo-Noise Was Added for Each Value of θ_U

End of Example 5.4

5.2.5 A Signal-Space Technique for Reducing Impact of Pointing Errors

The analysis of Section 5.2.3 suggests a more direct method of reducing the adverse impact of pointing error on the adaptive processor, one which does not require identifying an appropriate value for the parameter b in the norm bound (5.1), or of the pseudo-noise level μ_0 in (5.3).

Specifically, the useful component of the weight vector $\vec{w}$ is

$$T_0\vec{u}_0 = T_0[\vec{u}_0 + \delta\theta\ \vec{u}_0'] = \frac{1}{\pi_0}\vec{u}_0 + O(\delta\theta) \tag{5.4}$$

The matrix T_0 is the pseudo-inverse of the "signal part" of the covariance matrix R_{xx}. That is,

$$T_0 = (SPS^H)^{\#} \tag{5.5}$$

where # denotes the pseudo-inverse operation. The "signal part" of the covariance matrix R_{xx} can be calculated straightforwardly by eigenanalysis. Specifically,

$$SPS^H = E_s[\Lambda_s - \sigma^2 I]E_s^H \tag{5.6}$$

where Λ_s denotes the $(J+1)\times(J+1)$ diagonal matrix of the largest (signal-subspace) eigenvalues of R_{xx}, σ^2 is the value of the remaining $K-J+1$ (noise-subspace) eigenvalues, and E_s denotes the $K\times(J+1)$ matrix of signal-space eigenvectors. The required pseudo-inverse can be calculated as follows from the eigenvectors and eigenvalues of R_{xx}

$$T_0 = (SPS^H)^{\#} = E_s[\Lambda_s - \sigma^2 I]^{-1}E_s^H \tag{5.7}$$

The weight vector is then taken to be

$$\hat{\vec{w}} = T_0\vec{u}_0 = E_s[\Lambda_s - \sigma^2 I]^{-1}E_s^H\vec{u}_0 \tag{5.8}$$

The method is identical to the procedure proposed by Gabriel[5] who utilized the principal eigenvectors of the covariance matrix as a means for reducing the dimensionality of the adaptive nulling techniques. Gabriel pointed out the advantages of utilizing these principal eigenvectors to reduce noisy sidelobes and to achieve a tailored beam response. However, Gabriel did not address the advantages of this method with respect to pointing errors when the user signal is present.

In the following example, we demonstrate that, *for the steady-state case*, the signal-space method is effective in reducing the effects of user pointing errors.

Example 5.5: Signal Subspace Method

As an illustration of the signal-subspace method for reducing the effects of pointing errors, we consider again the scenario of Example 5.1. Curve 1 in Figure 5.6 summarizes the baseline performance of the adaptive processor as a function of user arrival angle θ_U when the steering vector is exactly matched to θ_U. Curve 2 depicts the performance when the user arrival angle varies and the steering vector is fixed to steer in the direction of the $\theta = 5°$. Curve 3 summarizes the performance of the signal-subspace method. The data for Curve 3 was obtained by determining the steady-state covariance matrix for each user arraival angle, and computing the matrix T_0 by performing and eigenanalysis and retaining the three principal eigenvectors. As can be seen, retaining the signal portion of the covariance matrix and discarding the noise portion substantially decreases the sensitivity of the processor to errors in pointing angle.

End of Example 5.5

5.2.6 Steady State Performance of the New Algorithms Against Mismatch

As a consequence of removing the user signal prior to weight determination, the new class of algorithms described in Section 3.5 is comparatively ***insensitive to errors in the assumed user direction.*** Specifically, if a small error is made in the pointing direction, then the pre-filter having weight set $\vec{w}_{\overline{U}}$ does not completely remove the user signal; rather, it passes the signal at an attenuated level. If the attenuated user signal does not significantly exceed the noise floor, then the weight vector $\vec{w}_J$ determined by the follow-on processor does not null the user. The following example illustrates this point.

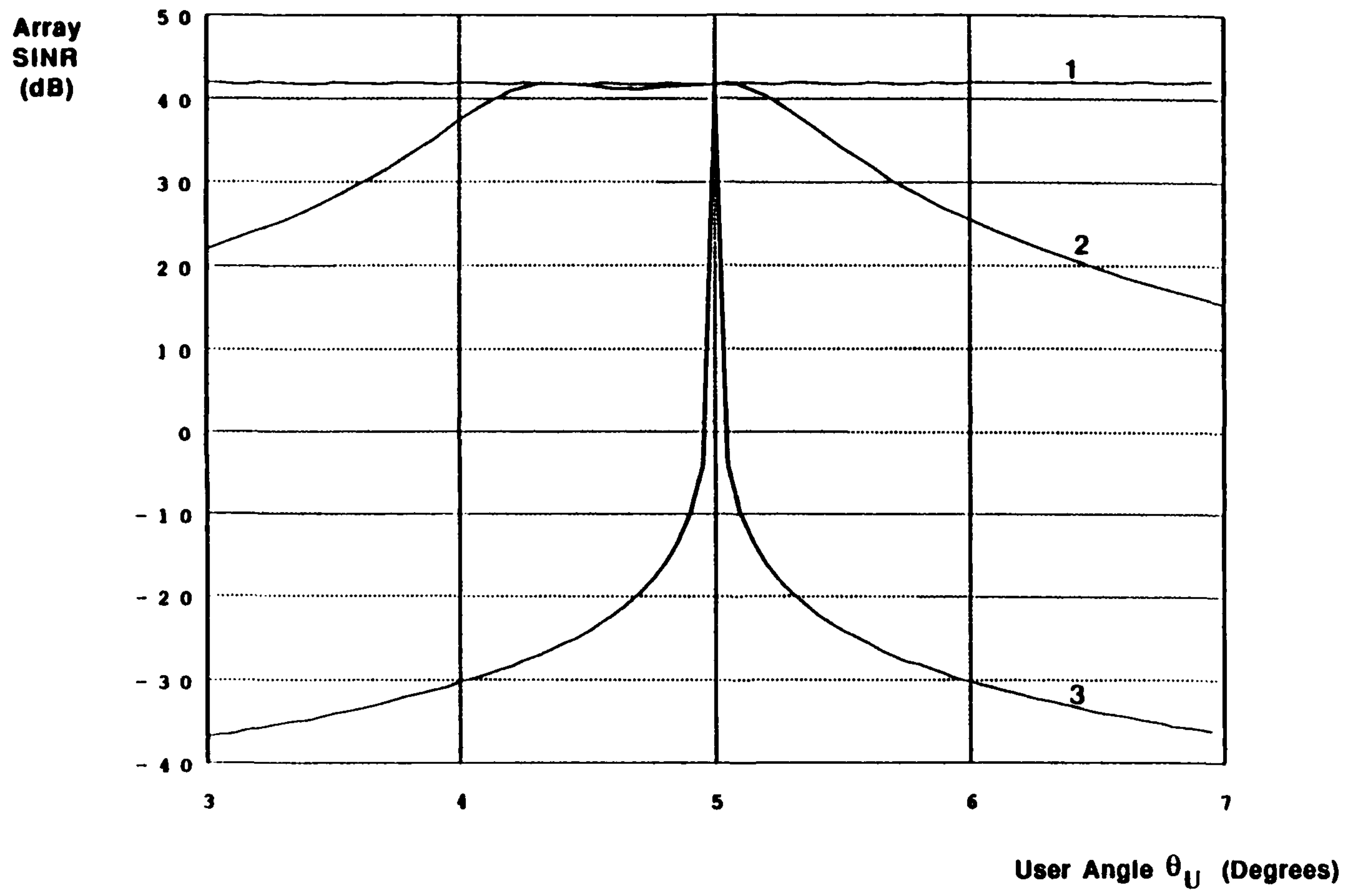
Example 5.6: New Algorithm with Two-Point Pre-Filter

Consider again the scenario of Example 5.1. An example algorithm was constructed from the class described in Section 3.3 as follows:

1. The weight set $\vec{w}_{\overline{U}}$ was selected to be a two-point filter (dipole) that places a null in the assumed direction of the user (i.e., $\theta = 5°$).

2. The matrix M was taken to be the identity matrix.

Figure 5.6: Use of Signal-Subspace Algorithm to Reduce Impact of Pointing Errors (Example 5.5)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector
3. SINR as a Function of User Position for Fixed 5 Degree Steering Vector Using Signal-Subspace Algorithm

16-Element Linear Array
30 dB SNR User Signal Present
at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

3. The weight set $\vec{w}_U$ was selected to be a two-point filter matched to the assumed direction of the user (i.e., which steered a beam in the assumed direction of the user).

4. The antenna pattern for the final weight vector $\vec{w}$ was constrained to provide unity gain in the direction of the user.

The sensitivity of the algorithm to pointing errors was examined for the case of steady state performance by using the (steady-state) matrix R_{xx} as opposed to the estimated covariance matrix $\hat{R}_{xx}$. The SINR of the composite adaptive processor was calculated for a range of values for the (true) user angle θ_U. The results are presented in Figure 5.7a. Curve 3 depicts the output SINR vs. θ_U. Curves 1 and 2 are the curves of Example 5.1, denoting respectively the performance of conventional SMI for the case of no mismatch, and mismatch of $(\theta_U - 5°)$. Clearly, the algorithm is significantly less sensitive than conventional SMI to pointing errors.

Figure 5.7b depicts the array signal-to-noise ratio (ASNR) at the output of the top filter of Figure 3.14 as a function of the true user angle θ_U. Here, the ASNR was defined as follows:

$$\text{ASNR} = \frac{\text{Output signal power for weight vector } w_J \text{ matched to residual user signal}}{\text{Output noise power for weight vector } w_J \text{ matched to residual user signal}} \quad (5.9)$$

Comparison of Curve 3 of Figure 5.7a with the curve of Figure 5.7b shows that Curve 3 begins to fall away from the no mismatch value at $\theta_U = 5°$ as the ASNR of the residual user signal begins to exceed 0 dB.

End of Example 5.6

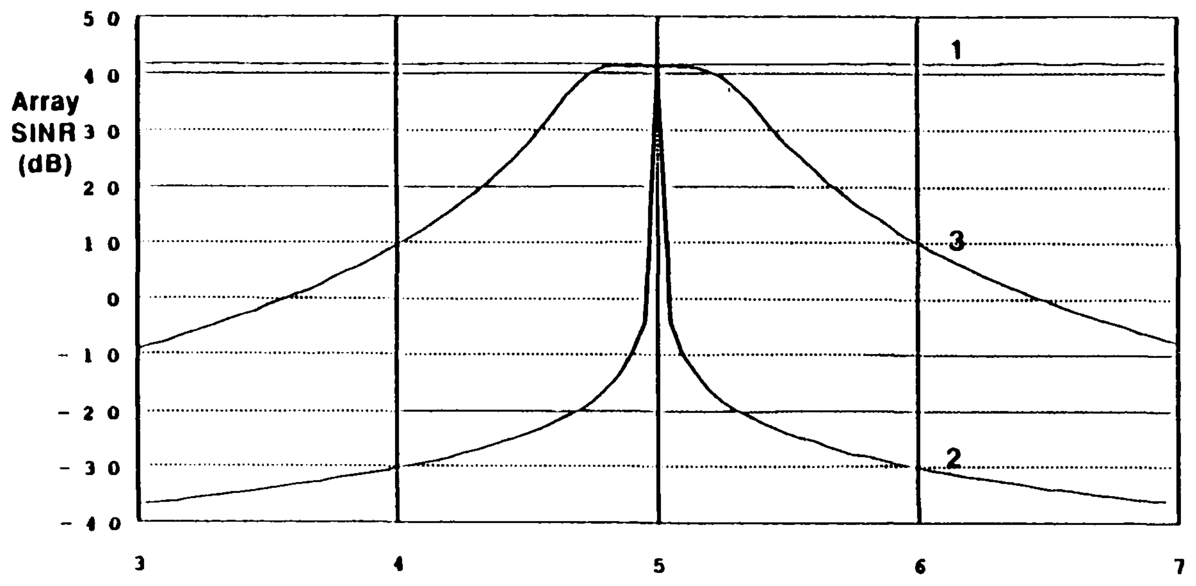
Given that performance for the composite filter of Figure 3.14 begins to depart from the no-mismatch case when the user signal output by the pre-filter with weight set $\vec{w}_{\overline{U}}$ is comparable to the noise output, an obvious way to further reduce sensitivity is to ***design the pre-filter so as to suppress the user signal over a wider angular sector***. The following example illustrates this approach.

Example 5.7: New Algorithm with Three-Point Pre-Filter

Consider again the scenario of Example 5.1. An alternative algorithm was constructed by specializing the processor of Section 3.3 as follows:

1. The weight set $\vec{w}_{\overline{U}}$ was selected to be a three-point filter that places a double null in the assumed direction of the user (i.e., $\theta_U = 5°$).

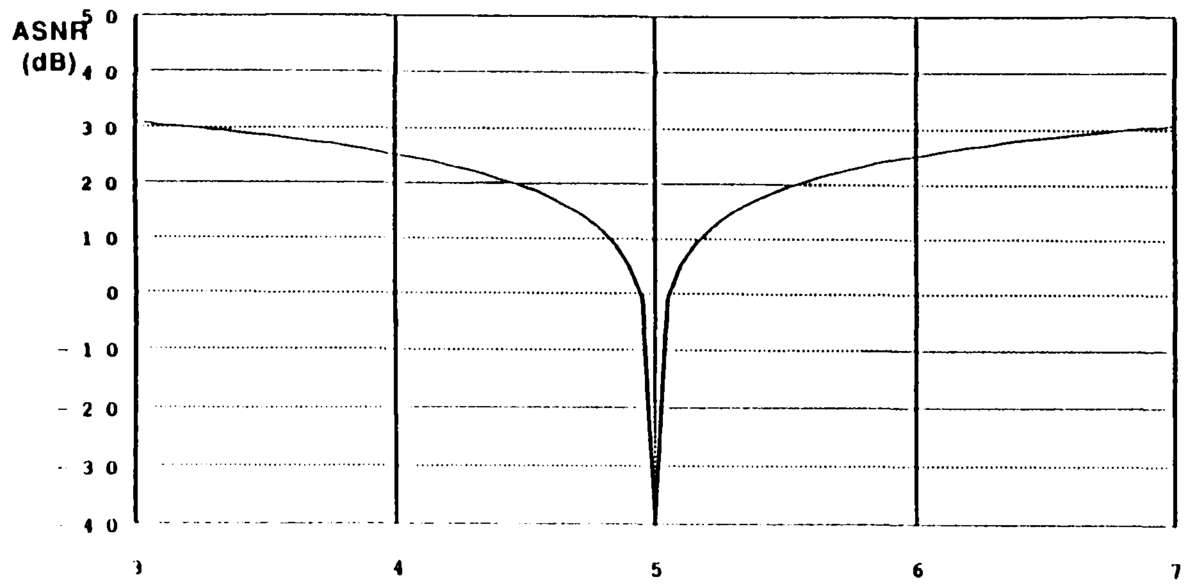
Figure 5.7a: Use of New Algorithm With Two-Point Pre-Filter to Reduce Impact of Pointing Errors (Example 5.6)



CURVES:
1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector
3. SINR as a Function of User Position for Fixed 5 Degree Steering Vector. Using New Algorithm With Two-Point Pre-Filter That Places a Null in Assumed User Direction of $\theta = 5°$

User Angle θ_U (Degrees)

Figure 5.7b: Pre-Filter ASNR As A Function of User Angle for Two-Point Pre-Filter (Example 5.6)



16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U In the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

User Angle θ_U (Degrees)

2. The matrix M was taken to be the identity matrix.
3. The weight set $\vec{w}_U$ was selected to be a three-point filter matched to the assumed direction of the user.
4. The antenna pattern of the final weight vector $\vec{w}$ was constrained to provide unity gain in the assumed direction of the user.

The steady-state SINR of the composite processor again was calculated for a range of values of θ_U. The results are presented in Curve 4 in Figure 5.8a. Curves 1, 2 and 3 are the curves of Figure 5.7a repeated for comparison. Clearly, the expanded pre-filter desensitizes performance to mismatch over a significantly wider angular sector than the two-point filter of Example 5.6 (Curve 3). Figure 5.8b presents a plot of the ASNR versus θ_U for the pre-filter. Comparison of Figures 5.5a and 5.5b shows once again that performance of the composite processor departs from the no-mismatch value when the pre-filter ASNR begins to significantly exceed 0 dB.

End of Example 5.7

Yet another method for reducing sensitivity of performance to pointing errors is to utilize the algorithm class of Section 3.3 in conjunction with one or more derivative constraints. The following example illustrates the opportunities.

Example 5.8: New Algorithm with Gain and Derivative Constraints

Consider again the scenario of Example 5.1. The algorithm class of Section 3.3 was specialized as was done in Example 5.6 except that Step 4 was replaced by the following step

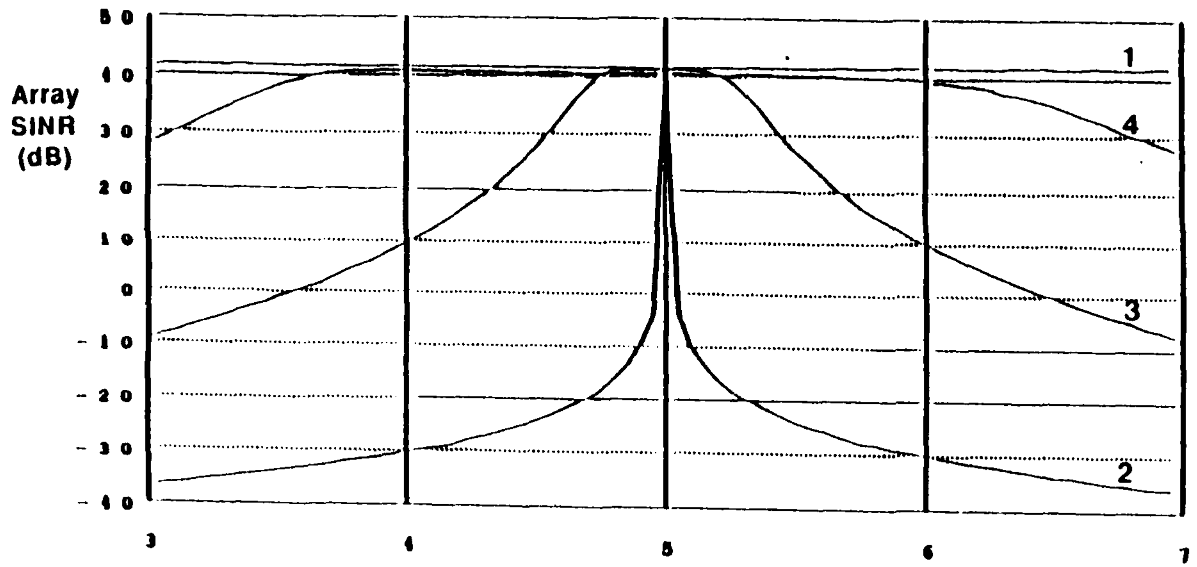
4) The antenna pattern of the final weight vector $\vec{w}$ was constrained to provide unity gain *and zero first derivative* in the assumed direction of the user.

Figure 5.9 presents the results of the pointing error experiments. Curve 4 depicts the SINR vs. θ_U behavior of the foregoing algorithm. Curves 1-3 are the curves of Example 5.7 repeated for comparison. Clearly, use of the derivative constraint provides a substantial reduction in sensitivity to pointing errors.

End of Example 5.8

The class of algorithms described in Section 3.3 was originally developed to accelerate the *transient response* of an adaptive processor in dynamic electromagnetic environments. It

Figure 5.8a: Use of New Algorithm to Reduce Impact of Pointing Error

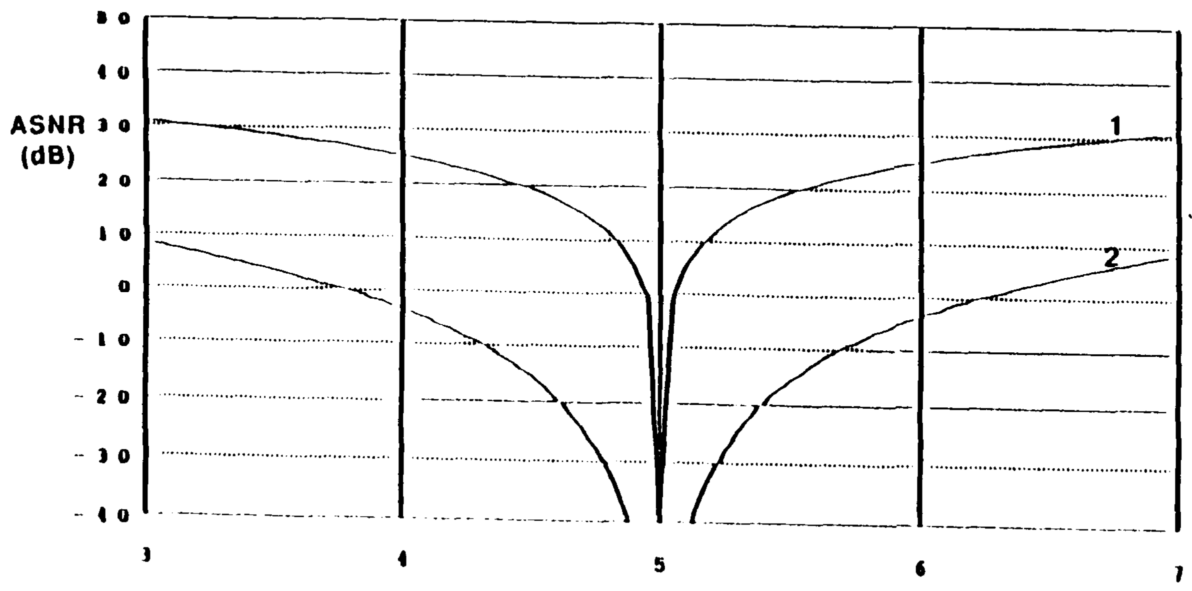


CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector
3. SINR as a Function of User Position for Fixed 5 Degree Steering Vector. Using New Algorithm With Two-Point Pre-Filter
4. SINR as a Function of User Position for Fixed 5 Degree Steering Vector Using New Algorithm With Three-Point Pre-Filter

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

Figure 5.8b: Pre-Filter ASNR As A Function of User Angle

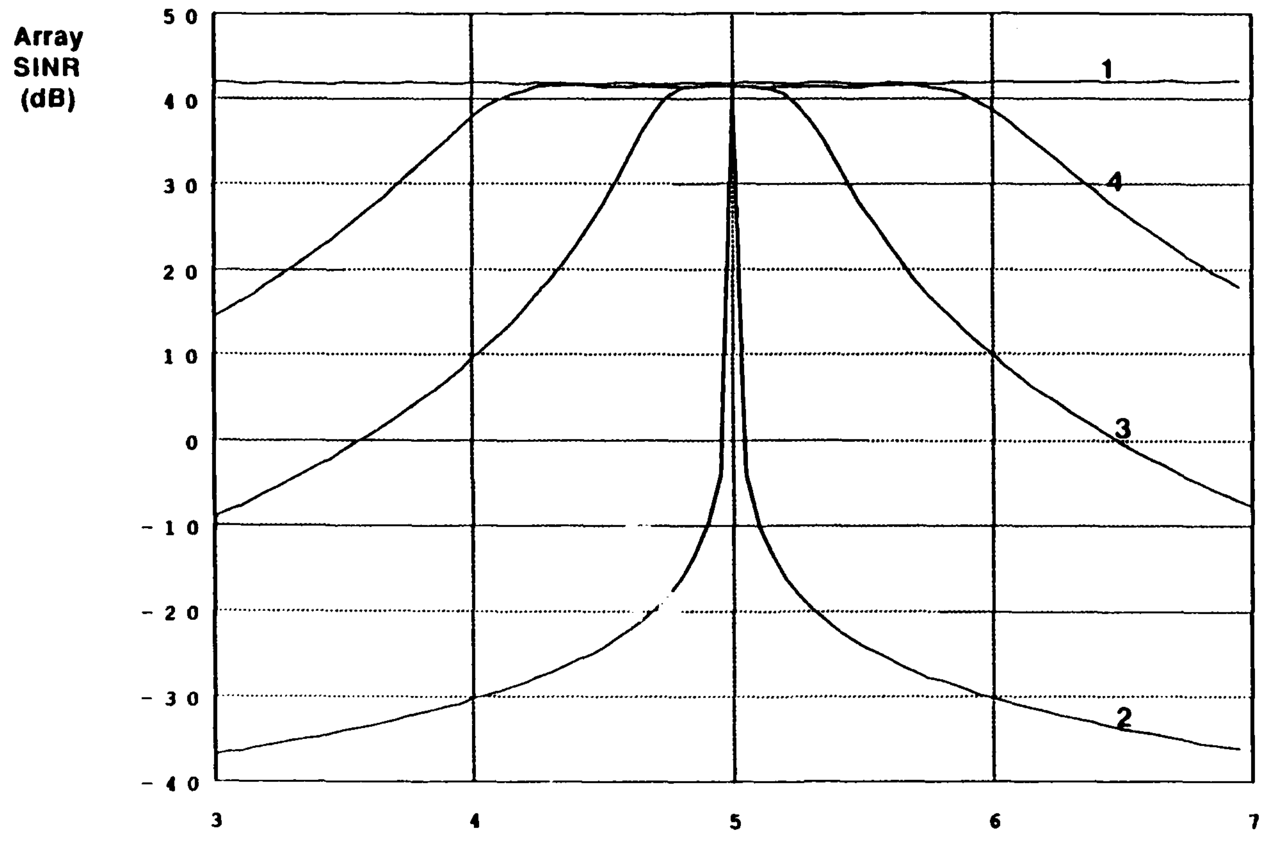


CURVES:

1. ASNR for Two-Point Pre-Filter
2. ASNR for Three-Point Pre-Filter

User Angle θ_U (Degrees)

Figure 5.9: Use of New Algorithm With Additional Derivative Constraint to Reduce Impact of Pointing Error (Example 5.8)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector
3. SINR as a Function of User Position for Fixed 5 Degree Steering Vector. Weights Computed Using New Algorithm With Gain Constraint
4. SINR as a Function of User Position for Fixed 5 Degree Steering Vector. Weights Computed Using New Algorithm with Additional Mainbeam Derivative Constraint.

User Angle θ_U (Degrees)

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

is clear from the examples in this section that the algorithms provide additional significant benefit. Specifically, the algorithms *automatically* reduce the sensitivity of *steady-state* interference suppression to pointing errors. Indeed the algorithm performance is much less sensitive to pointing errors than conventional algorithms which approximate the solution of the Wiener-Hopf equation [e.g. SMI, LMS, and Applebaum-Howells processors].

Figure 5.10 compares the *steady-state* performance of several of the algorithms considered in the foregoing examples. It is evident from the figure that the steady-state performance of the new algorithm using a two-point prefilter appears comparable to that of the conventional algorithms with compensation for the user presence (Curves 1-4). *Additional* tailoring of the new algorithm in the form of more elaborate pre-filters and linear constraints further desensitizes the dependence of performance on pointing errors, (for example, see Curve 6).

5.2.7 Transient Performance With Pointing Errors

Whereas the methods of Sections 5.2.3-5.2.5 are effective in reducing the performance sensitivity of conventional nulling algorithms under *steady-state* conditions, *these measures are of limited utility under transient conditions.*

By contrast, the new class of algorithms described in Section 3.3 reduces performance sensitivity to pointing errors under *both steady-state and transient conditions.* The effective performance under transient conditions is due to excision of the user signal prior to weight determination.

The following examples illustrate these points.

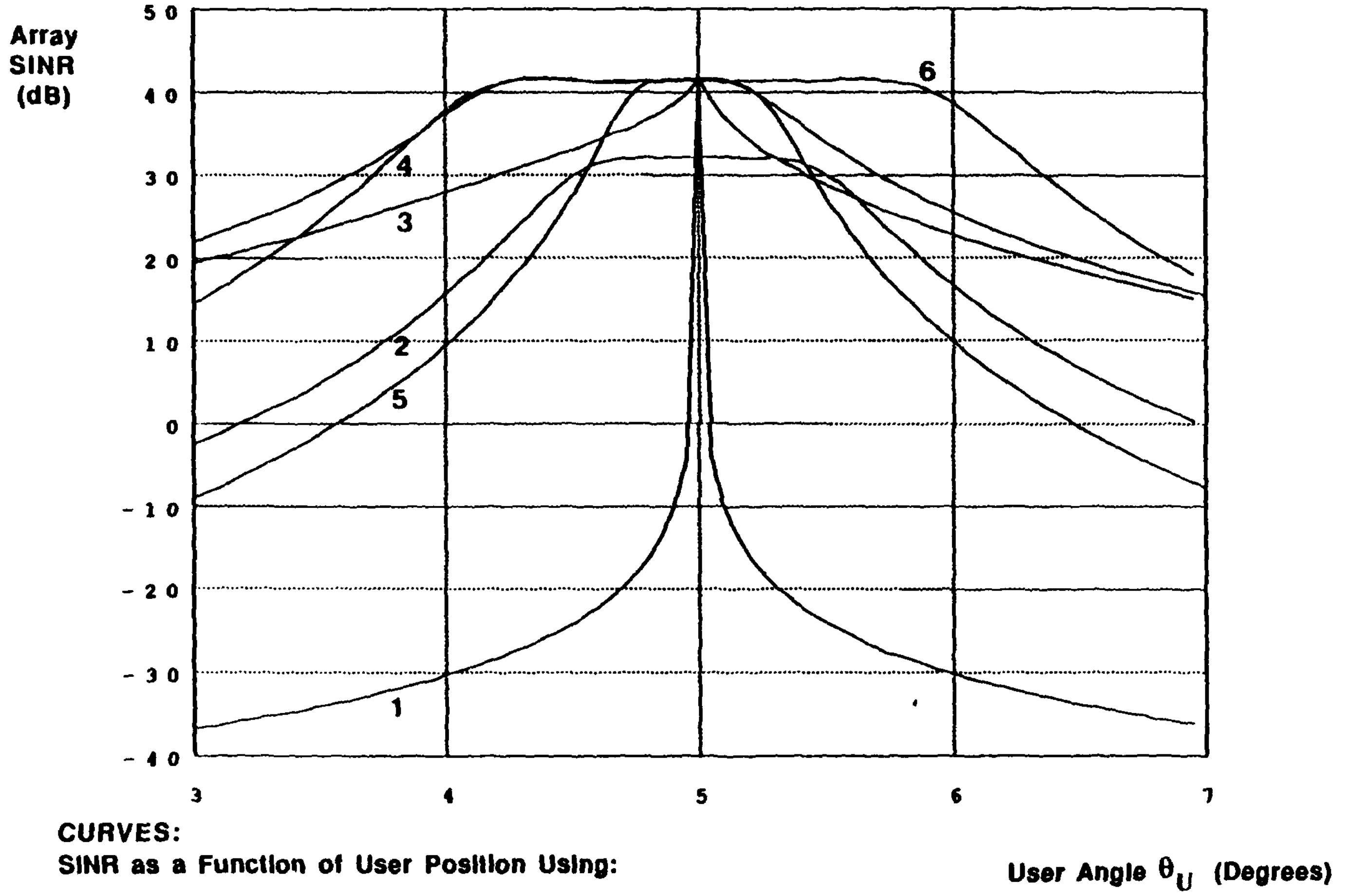
Example 5.9: Conventional Algorithms with Fixes After 100 *Snapshots*

A sample covariance matrix $\hat{R}_{xx}$ was constructed based upon 100 snapshots for the scenario of Example 5.1. The matrix then was used in place of the (steady-state) covariance matrix R_{xx} for the conventional algorithms of Examples 5.3 and 5.5.

Figure 5.11 illustrates the SINR vs. θ_U behavior of SMI when the user signal is present at 30 dB SNR, and derivative constraints are imposed. Curve 1 is the curve of Figure 5.1 repeated for reference. Curves 2, 3 and 4, respectively, depict the SINR vs θ_U for SMI when the antenna pattern for the composite processor is constrained to have:

- Unity gain in the assumed user direction of $\theta = 5°$.

Figure 5.10: Comparative Performance of Conventional Algorithms and New Algorithm Which Reduce Impact of Pointing Errors

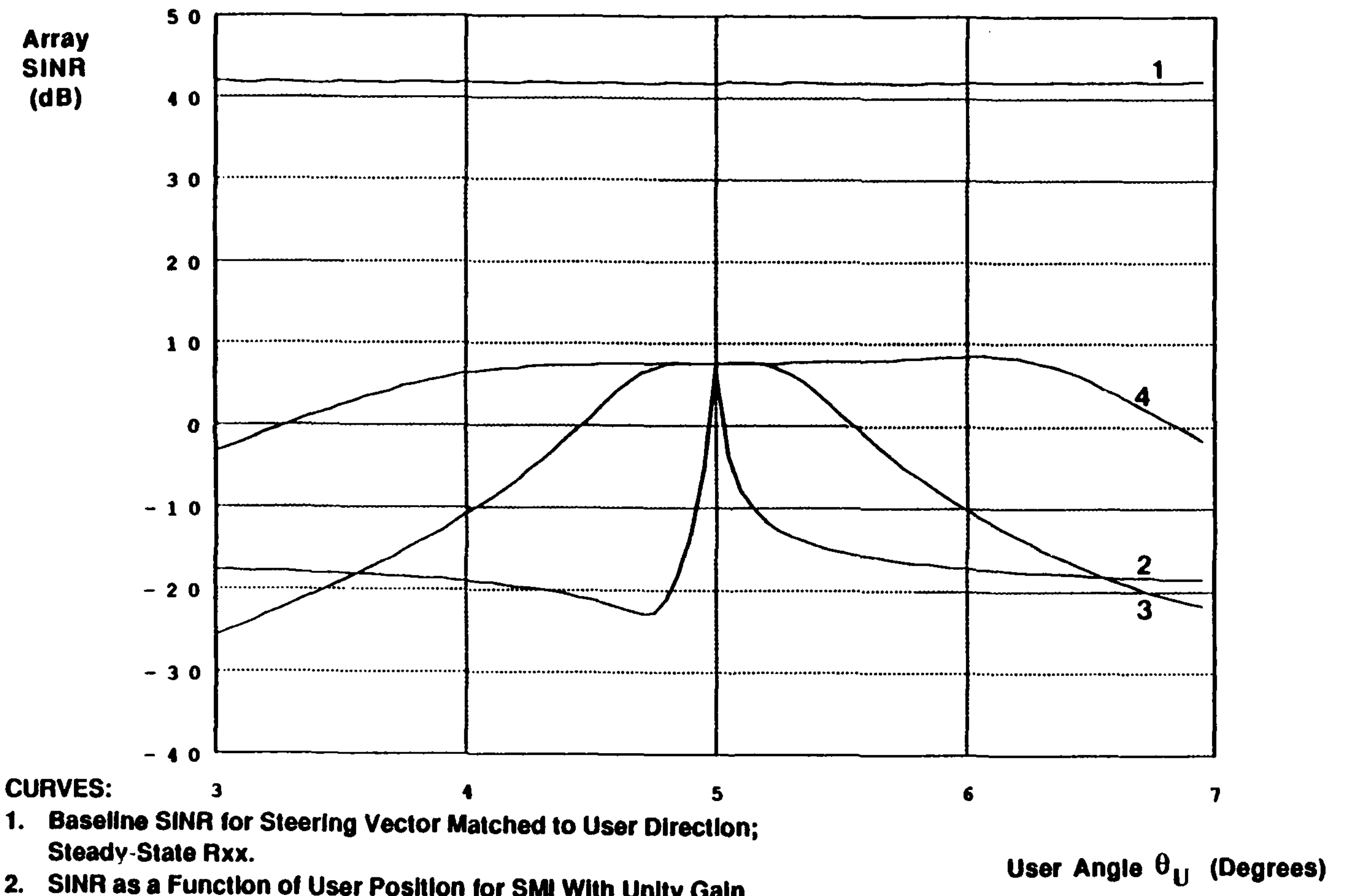


CURVES:
SINR as a Function of User Position Using:

1. SMI Algorithm
2. SMI Algorithm with Derivative Constraint
3. SMI Algorithm with Optimal Pseudo-Noise
4. Signal Sub-Space Algorithm
5. New Algorithm with Two-Point Pre-Filter
6. New Algorithm with Two-Point Pre-Filter and Derivative Constraint

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

Figure 5.11: Illustration of Transient Performance of Derivative-Constrained SMI With User Present and Pointing Error (Example 5.9)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction; Steady-State Rxx.
2. SINR as a Function of User Position for SMI With Unity Gain Constraint at $\theta = 5°$; $\hat{R}xx$ Computed Using 100 Snapshots.
3. SINR as a Function of User Position for SMI With Unity Gain and Zero First Derivative Constraint at $\theta = 5°$; $\hat{R}xx$ Computed Using 100 Snapshots.
4. SINR as a Function of User Position for SMI With Unity Gain and Zero First and Second Derivative Constraints at $\theta = 5°$; $\hat{R}xx$ Computed Using 100 Snapshots.

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

- Unity gain and zero first derivation in the assumed user direction.
- Unity gain and zero first and second derivatives in the assumed user direction.

It is clear from Curves 2, 3, and 4 that performance of SMI with derivative constraints is far below that depicted by Curve 1.

Figure 5.12 depicts the SINR vs. θ_U behavior of the "thinned-beam" variant of SMI described in Example 5.5 for the case of 30 dB SNR and 100 snapshots. Again, Curve 1 is identical to Curve 1 of Figure 5.1. Curve 2 depicts the SINR vs. θ_U behavior of "thinned-beam" SMI to mismatch of $(\theta_U - 5°)$. Clearly, the performance of the thinned-beam SMI technique after 100 snapshots is much poorer than the performance of the same method in the steady-state.

End of Example 5.9

Example 5.10: New Algorithms After 100 Snapshots

The same 100 snapshot sample covariance matrix $\hat{R}_{xx}$ was used to assess the sensitivity to pointing error of the (new) algorithms described in Examples 5.6 and 5.7.

Figure 5.13 compares the SINR vs. θ_U behavior of the Example 5.6 algorithm after 100 snapshots (Curve 3) with the steady-state performance (Curve 2). Clearly, performance for the 100 snapshot case is insensitive to small pointing errors, and is almost identical to that for the steady-state case.

Figure 5.14 depicts the SINR vs. θ_U behavior of the algorithm of Example 5.7 after 100 snapshots (Curve 3), and under steady-state conditions (Curve 2). Performance for the 100 snapshot case is insensitive to modest pointing errors, and almost is identical to that for steady-state conditions.

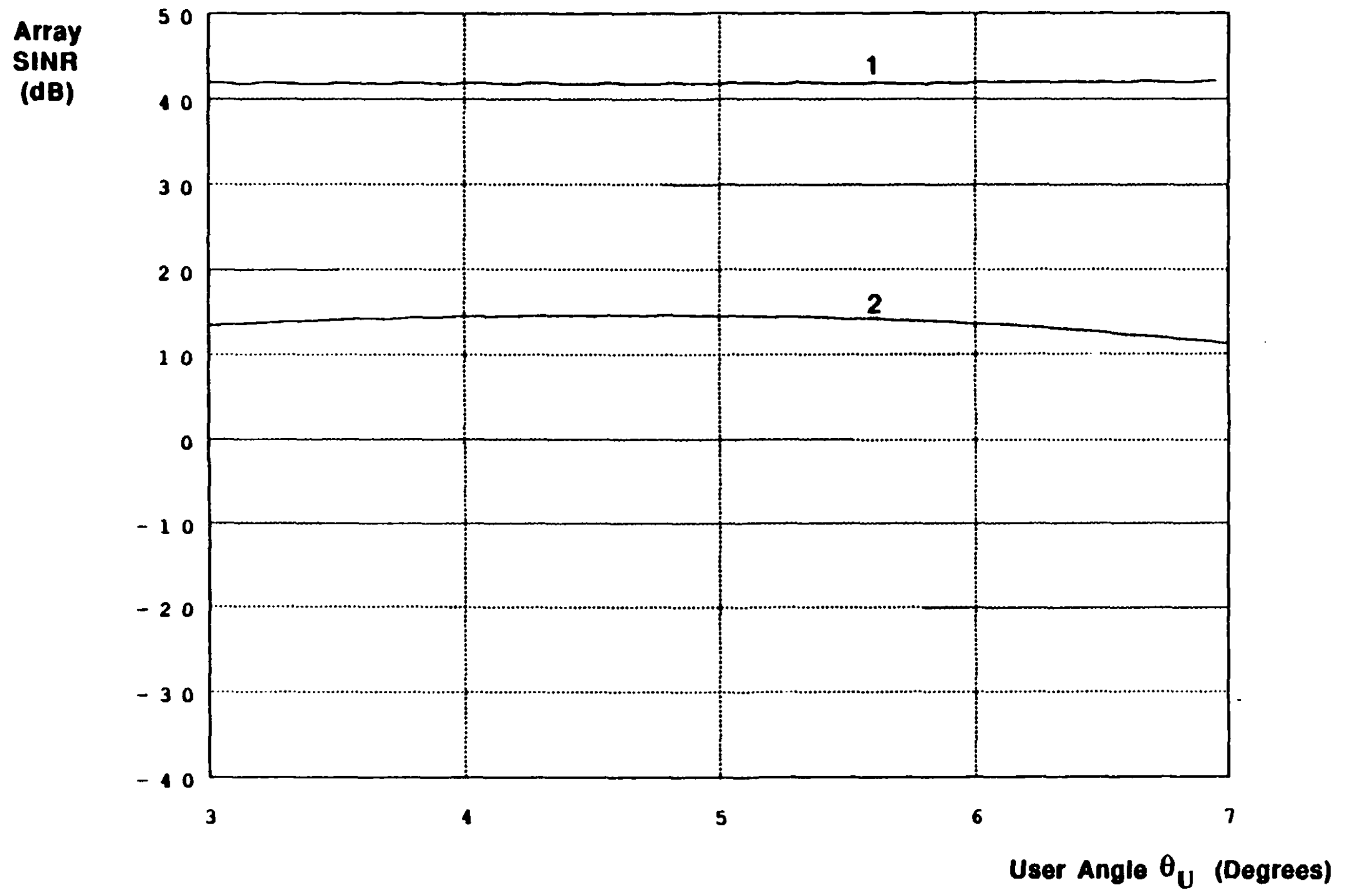
End of Example 5.10

5.3 Impact of User Signal Presence on Channel Hardware Errors

5.3.1 Introduction

The purpose of this section is to describe the channel hardware error problem in adaptive arrays and to review both classical and new methods for reducing the impact of this type

Figure 5.12: Illustration of Transient Performance of Signal Sub-Space Method with User Present and Pointing Error (Example 5.9)

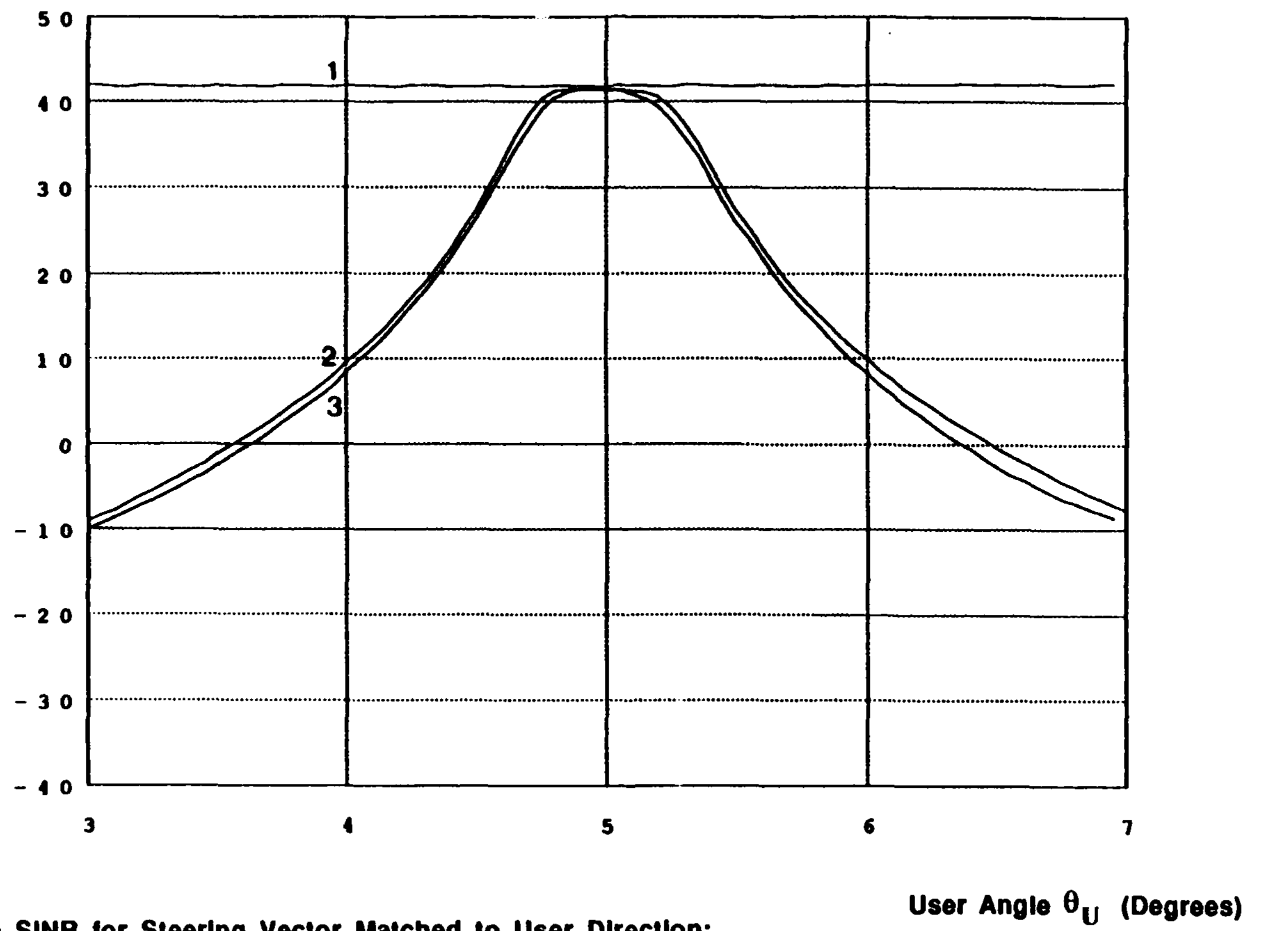


CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction; Steady-State Rxx.
2. SINR as a Function of User Position for Fixed 5 Degree Steering Vector Using Signal Subspace Algorithm; $\hat{R}xx$ Computed Using 100 Snapshots.

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

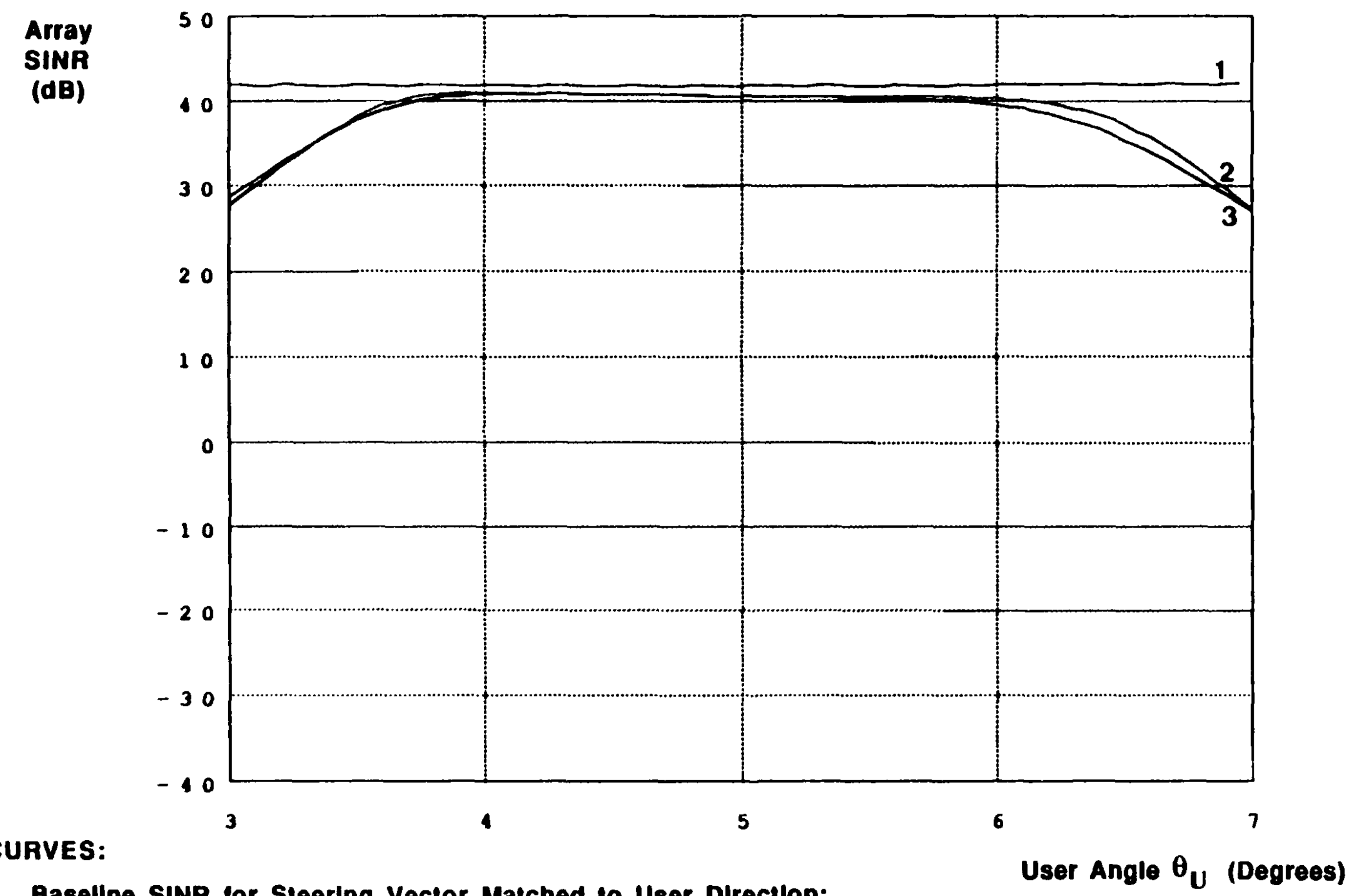
Figure 5.13: Illustration of Transient Performance of Example 5.6 Algorithm (Two-Point Pre-Filter)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction; Steady-State Rxx.
2. SINR as a Function of User Position Using Two-Point Pre-Filter Algorithm; Steady-State Rxx.
3. SINR as a Function of User Position Using Two-Point Pre-Filter Algorithm; 100 Snapshot $\hat{R}$xx.

Figure 5.14: Illustration of Transient Performance of Example 5.7 Algorithm (Three-Point Filter)



CURVES:

1. Baseline SINR for Steering Vector Matched to User Direction; Steady-State Rxx.
2. SINR as a Function of User Position Using Three-Point Pre-Filter Algorithm; Steady-State Rxx.
3. SINR as a Function of User Position Using Three-Point Pre-Filter Algorithm; 100 Snapshot $\hat{R}xx$.

16-Element Linear Array
30 dB SNR User Signal Present at Angle θ_U in the Adaptation Data
30 dB JNR Jammers at 15, 50 Degrees

of error.

5.3.2 Impact of Channel Hardware Errors on Performance

The previous section discussed the impact of user signal presence on the performance of some adaptive arrays when pointing errors were present. Presence of the user signal can also magnify the impact of channel hardware errors. In effect, channel hardware errors alter elements in the covariance matrix so that there is a mismatch between the user contribution to the covariance matrix and the steering vector $\vec{u}_0$. As discussed in [1][9][10], random hardware errors can be interpreted as random pointing errors which do not necessarily correspond to azimuth or elevation errors. Therefore, the adaptive processor once again mistakes the user signal for a mainbeam jammer, and attempts to null the user.

To clarify the foregoing point, assume that exact pointing information is available so that the user unit-arrival vector $\vec{u}_0$ is correct. Consider the impact of fixed channel-to-channel gain and phase errors.

Assume:

1. The weights are determined based upon measurements made in one set of channels which contain channel-to-channel gain and phase errors,
2. The resultant weights are applied to the *same* set of channels.

The covariance matrix $\tilde{R}_{xx}$ for the channels with hardware errors then can be expressed in terms of the exact covariance matrix R_{xx} as follows

$$\tilde{R}_{xx} = G R_{xx} G^H \tag{5.10}$$

where

$$G = I + \Lambda_\epsilon \tag{5.11}$$

and Λ_ϵ is a complex-valued diagonal matrix which contains the random gain and phase errors.

The solution of the exact Wiener-Hopf equation using $\tilde{R}_{xx}$ in place of R_{xx} then is

$$\vec{w} = \mu \tilde{R}_{xx}^{-1} \vec{u}_0 \tag{5.12}$$

Thus,

$$\vec{w} = \mu (G^H)^{-1} R_{xx}^{-1} G^{-1} \vec{u}_0 = \mu (G^H)^{-1} R_{xx}^{-1} \tilde{\vec{u}}_0 \tag{5.13}$$

where

$$\tilde{\vec{u}}_0 = G^{-1}\vec{u}_0 \tag{5.14}$$

The matrix factor within the brackets of (5.13) is the inverse of the exact covariance matrix based upon the correct unit arrival vectors. However, the factor $\tilde{\vec{u}}_0$ is distorted or mismatched relative to $\vec{u}_0$. Thus, the impact of the channel errors is quite similar to that of pointing errors. The following examples illustrate this point:

Example 5.11: Wiener-Hopf (Steady-State) Solution in the Presence of Hardware Errors

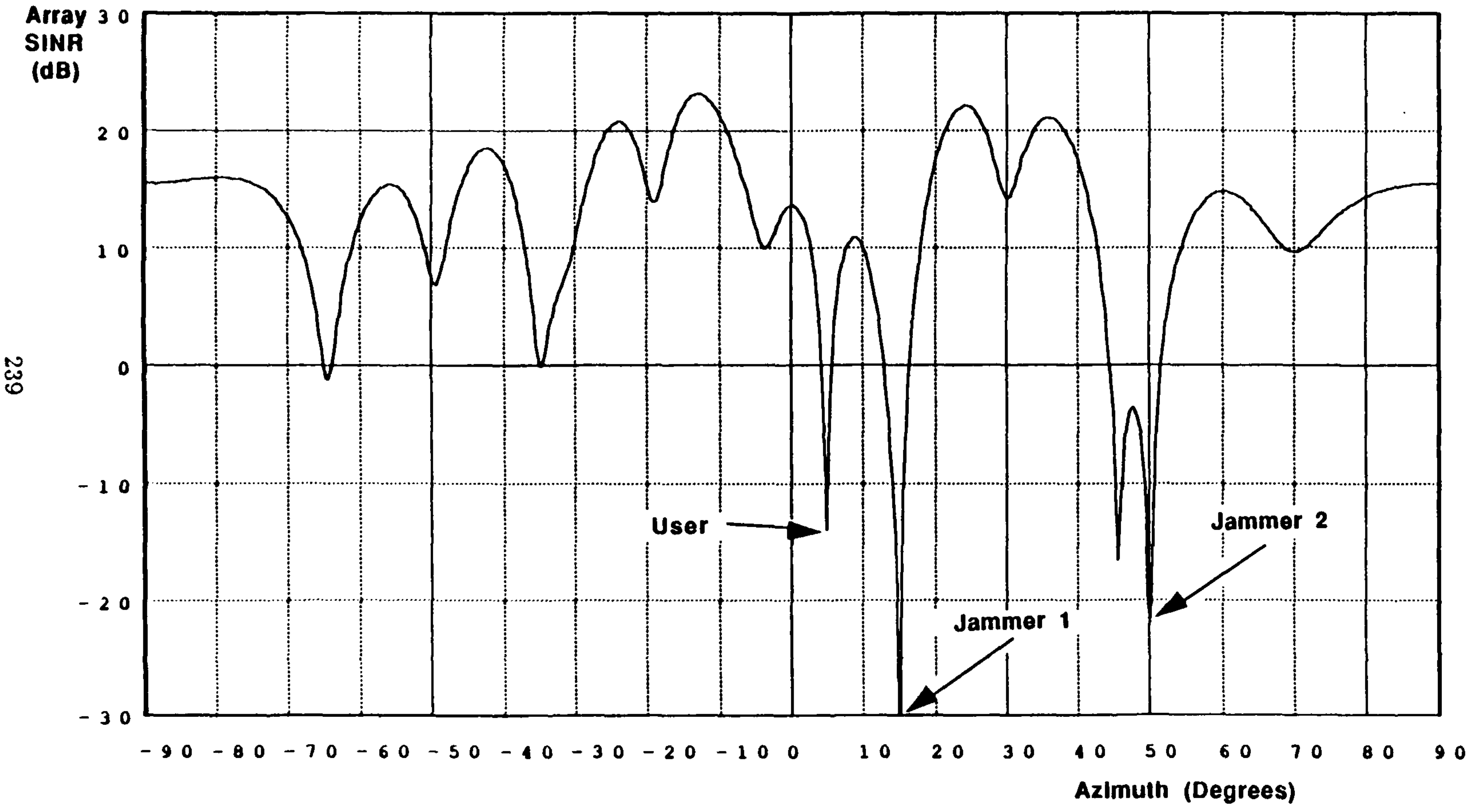
We consider again the scenario of a 16-element narrowband linear array in the presence of a 30 dB JNR jammer at $\theta = 15°$ and a 30 dB jammer at angle $\theta = 50°$. The 30 dB user is located at an angle of $\theta = 5°$.

To simulate the effects of channel hardware errors, we perturbed the gains and/or the phase shifts applied to each channel. The gain and phase perturbations were derived as follows. For gain variations of $x\%$ we determined a random set of numbers that varied uniformly over the range of $1 \pm (x/100)$. This random set of gains was applied to the set of hardware channels so that the covariance matrix contained the effects of channel-to-channel gain errors. Similarly, for phase variations of y-degrees we determined a random set of numbers that varied uniformly over the range of $0.0 \pm y$. This random set of phase shifts was applied to the set of hardware channels so that the covariance matrix contained the effects of channel-to-channel phase errors. We refer to the "level" of hardware errors as $x\%$ gain errors and y-degree phase errors.

To simulate the effects of hardware errors in this example, we assumed values of 1% gain errors (≈ 0.1 dB) and 4° phase errors. We assume, in this example, that an infinite number of snapshots are available for determining the weights, and that the direction of the user is known exactly. However, we also assume that the steering vector $\vec{u}_0$ points toward 5° and has not been adjusted in order to compensate for channel-to-channel hardware errors. Thus, degradation in array performance can be attributed solely to the mismatch between the data used to form the covariance matrix and the steering vector, and not to transient or pointing error effects.

The Wiener solution for the weights was computed, and the associated SINR was determined to be -12.5 dB. This is to be contrasted with an SINR of 41.8 dB which is obtained when the user is not present in the data used for adaptation. The beampattern for the weight set determined when the user is present in the face of the channel hardware errors is shown in Figure 5.15. The high sidelobes and the rapid variation in the beampattern

Figure 5.15: Beam Pattern of Adapted Weight Set With Uncompensated Channel Hardware Errors (1% Gain, 4° Phase) (Example 5.11)



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 50 Degrees

near the user direction of 5° is evident in the pattern. Once again the high sidelobes act to amplify the thermal noise of the array, thereby degrading the SINR.

In fact, this behavior is again associated with the placement of a null by the adaptive algorithm at a location *nearby* the user. To see this more clearly, we show in Figure 5.16 the zero pattern of the steady-state weight set after adaptation, with hardware errors, when the 30 dB user signal is present. The gain constraint forces the weight set to yield unity gain at the location $1.0 \exp(j\pi \sin 5) = 1 \exp(j.27381)$ while the adaptive weights placed a zero at location $.99393 \exp(j.272208)$. The placement of the zero is due to the uncompensated hardware errors which displace the actual user arrival data away from the assumed user direction. In addition, the steering constraint no longer provides unity gain in the 5° direction since the weights are applied to channels with gain and phase errors.

The primary difference between the effects of pointing errors and channel hardware errors is that the former causes the zero to be placed on the unit circle, while the latter causes the zero to be placed near the unit circle.

End of Example 5.11

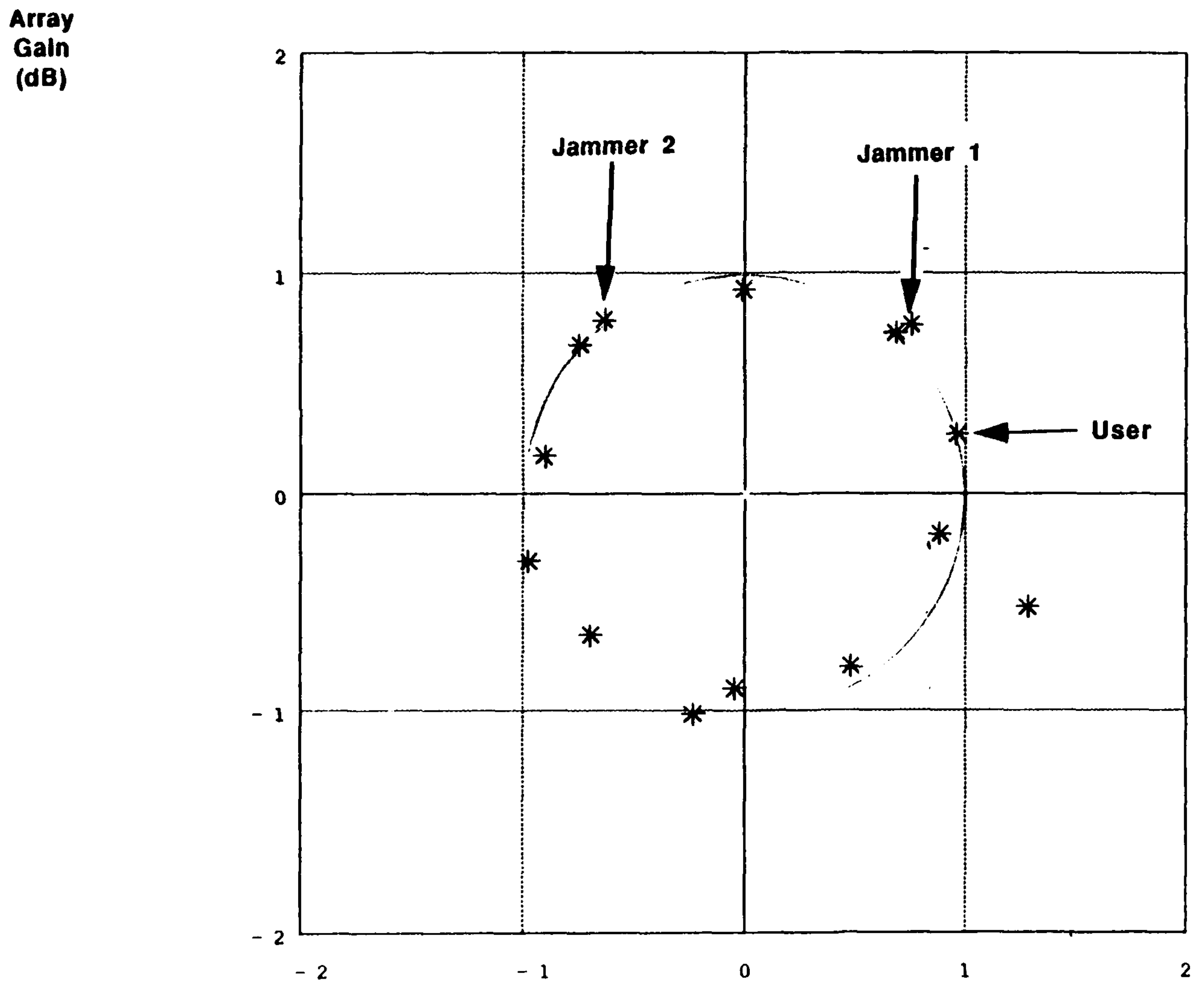
In general, the user signal acts to *magnify* the impact of hardware errors. The stronger the user signal, the greater the sensitivity to hardware errors. The following example illustrates this point.

Example 5.12: Steady-State Performance as a Function of User Level

To illustrate the importance of the user signal level on the impact of hardware errors, we used a 16-element array with 30 dB JNR jammers at $\theta = 15°$ and $\theta = 50°$. For each of a series of user levels, the gain errors were varied over the range of $0° - 4°$ (with no phase errors) and the phase errors were varied over the range from $0 - 2\%$ (≈ 0.2 dB, with no gain errors). For each value of hardware error, the steady-state adaptive weight solution was determined and the associated array gain was determined. Array gain is defined as array output SINR normalized by the single sensor SNR. The departure of the array gain from $10 \log_{10} 16 = 11.8$ dB is a measure of the impact of the user signal level on hardware errors.

Figure 5.17a illustrates the array gain as a function of hardware gain errors for differing levels of user signal in the adaptation data. Curve 1 corresponds to a user signal level of -10 dB SNR. Curves 2-5 correspond to successively higher levels of the user signal by 10 dB for each curve, i.e. Curve 5 corresponds to an SNR of 30 dB. Similarly, Figure 5.17b illustrates the sensitivity to phase errors for varying levels of user signal. The curves in Figures 5.17a and 5.17b indicate that when the user signal level significantly exceeds the thermal noise

Figure 5.16: Location of Zeros in Complex Z-Plane with Hardware Errors (1% Gain, 4° Phase) and 30 dB SNR User Present (Example 5.11)



level, and there exists mismatch between the covariance matrix and the steering vector due to channel hardware errors, the performance of the adaptive array decreases substantially.

It should be emphasized that in this example the adaptive weights are applied to the *same* set of channels as those used to compute the covariance matrix. Thus, the weight set is truly minimizing the output power of the array, subject to the steering vector constraint. The effect of hardware errors is to cause a mismatch between the constraint condition and the actual user contribution to the data. The degradation in performance can be attributed solely to a user signal effect. In a later example (Example 5.19), when the weights are computed from one set of channels, and ***applied to a different set of channels***, such as might be the case in a hybrid analog/digital system, a second effect is present. Namely, the mismatch between the measurement channels and application channels causes a limitation in the depth of the achievable nulls, thereby limiting nulling performance even when the user signal is not present in the adaptation data.

End of Example 5.12

5.3.3 Steady-State Performance of Classical Techniques for Reducing Impact of User Signal

A number of conventional methods have been proposed to overcome the effects of channel hardware errors. The following two examples indicate the general performance of these techniques.

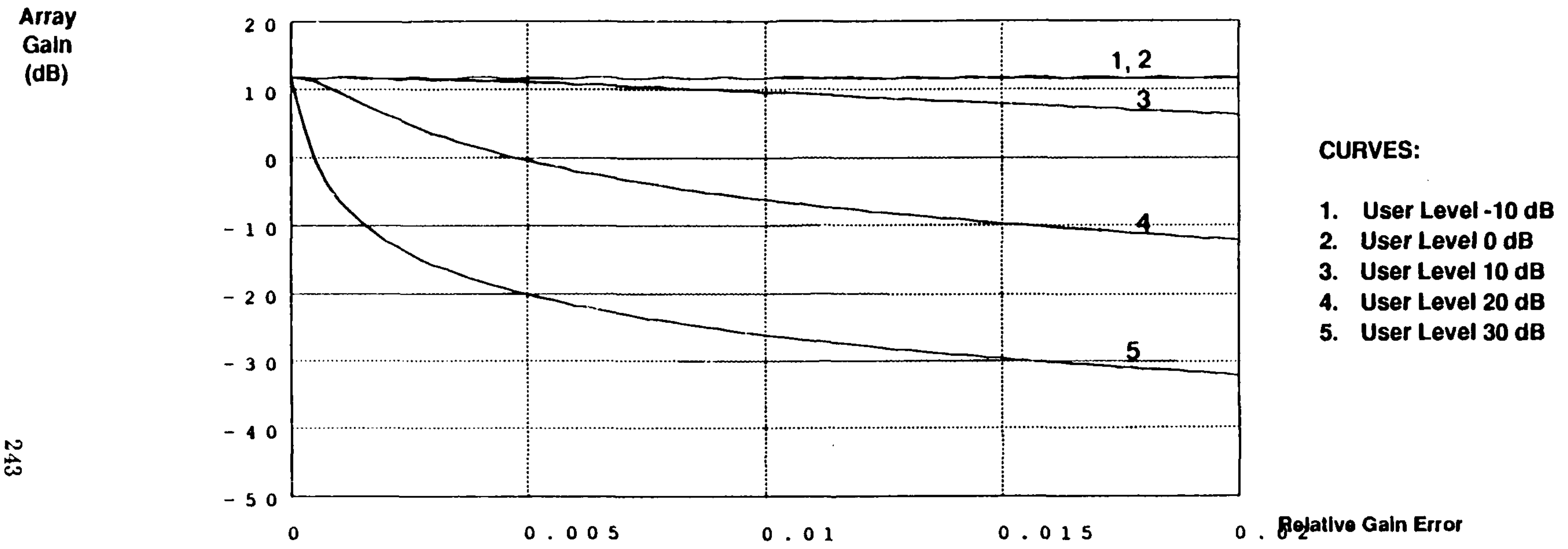
Example 5.13: Use of Pseudo-Noise to Reduce the Impact of Channel Hardware Errors on Steady-State Performance

The introduction of pseudo-noise into the covariance matrix provides one means for reducing the effects of channel hardware errors when the user signal is present. The effect is to increase the noise floor thereby increasing the output power of the array when the weights are large. Thus, there is an additional penalty due to large weight norm associated with attempting to suppress the user signal due to mismatch caused by hardware errors.

To illustrate the use of pseudo-noise as a mechanism for reducing the effects of channel hardware errors, we consider again the scenario in Example 5.11. The hardware errors were selected to be 1% (0.1 dB) gain errors and 4° phase errors. Figure 5.18 depicts the SINR as a function of the level of pseudo-noise (relative to the thermal noise level) added to the covariance matrix diagonal. The curve shows that the SINR increases from a value of -12.5

Figure 5.17: Array Gain as a Function of User Signal Level in the Presence of Hardware Errors (Example 5.12)

A



B

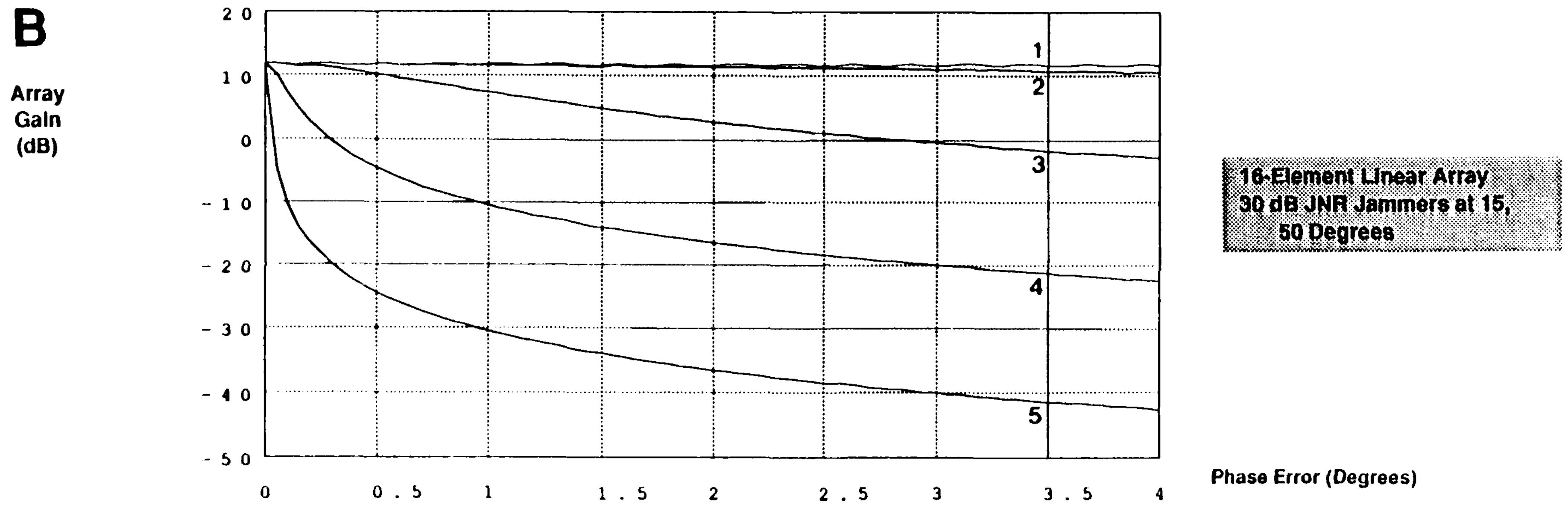
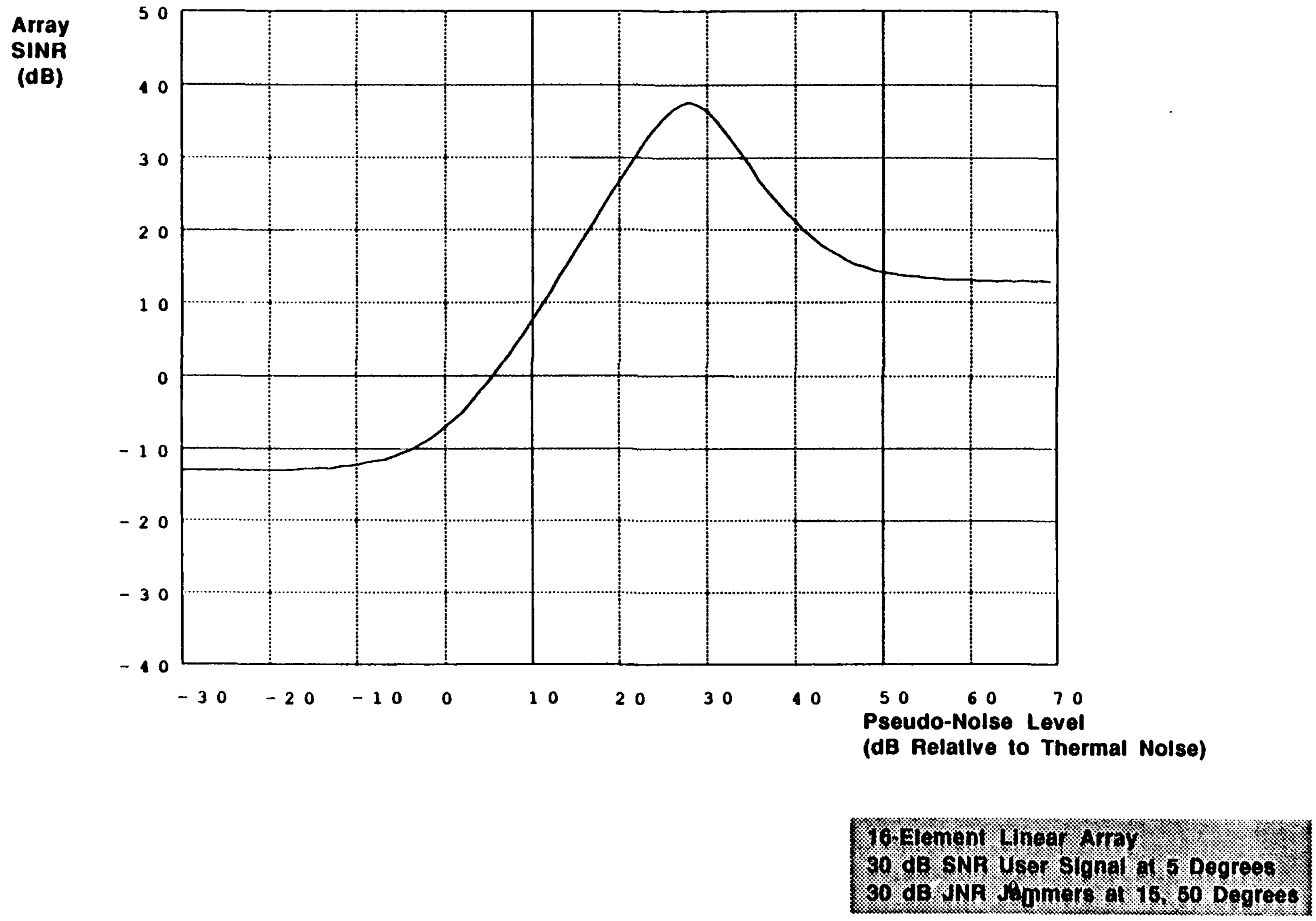


Figure 5.18: Addition of Pseudo-Noise to Improve Steady-State Performance for a Fixed Set of Hardware Errors (1% Gain, 4° Phase) (Example 5.13)



dB up to a peak value of 37 dB, corresponding to the optimal level of pseudo-noise, and then decreases to settle out at the final asymptotic value of 12.5 dB. An implication is that selection of the proper level of pseudo-noise provides steady-state performance improvement when channel hardware errors are present.

This effect is very similar to that obtained from using using pseudo-noise to reduce the effects of pointing errors. The advantage of the pseudo-noise method is that it provides reduced sensitivity to channel hardware errors. The disadvantage of the method is that a *different* level of pseudo-noise must be determined based on the magnitude of the hardware errors and the user/jammer scenario. Algorithms for determining the optimal level of pseudo-noise for a given set of channel hardware errors and scenario were not further explored in this work.

End of Example 5.13

Example 5.14: Use of the Signal Subspace Technique to Reduce the Impact of Channel Hardware Errors on Steady-State Performance

The utilization of the signal subspace provides another means for reducing the impact of channel hardware errors when the user signal is present. As described earlier, the signal subspace method is a beamspace technique which uses beams comprised of the principal eigenvectors of the covariance matrix. The signal subspace technique is identical to the procedure proposed by Gabriel[5] who utilized the principal eigenvectors of the covariance matrix as a means for reducing the dimensionality of the adaptive nulling techniques. Gabriel pointed out the advantages of utilizing these principal eigenvectors to reduce noisy sidelobes and to achieve a tailored beam response. However, Gabriel did not address the advantages of this method with respect to channel hardware errors when the user signal is present.

The signal subspace algorithm algorithm consists of determining the principal eigenvectors of the covariance matrix (i.e. those eigenvectors associated with eigenvalues significantly above the noise floor) and using then using these as beams. Both the covariance matrix and the steering vector are projected into beamspace using these beams. The beamspace Wiener weights are then determined and reprojected back to element space.

To illustrate the use of the signal subspace technique as a means for reducing the impact of channel hardware errors, we consider the scenario of Example 5.12 with the 30 dB SNR user signal present at 5°, and 30 dB JNR jammers present at $15^\circ, 50^\circ$. The gain errors were varied over the range of $0^\circ - 4^\circ$ (with no phase errors) and the phase errors were varied over the range from $0 - 2\%$ (≈ 0.2 dB). For each set of hardware errors, the steady-state adaptive weight solution was determined and the associated array output SINR was determined. The adaptive weights were determined using both conventional SMI and the signal subspace technique.

Figure 5.19a illustrates the steady-state performance of SMI (Curve 1) and the signal subspace technique (Curve 2) as a function of gain errors (with no phase errors). As can be seen, the performance of SMI deteriorates rapidly as the level of gain error increases thereby causing mismatch between the user steering vector and the user contribution to the covariance matrix. The signal subspace method substantially improves the steady-state performance with user signal present and channel gain errors.

Similarly, Figure 5.19b illustrates the performance of conventional SMI (Curve 1) and the signal subspace technique (Curve 2) as a function of channel phase errors (with no gain errors). As can be seen, the signal subspace method substantially improves the steady-state performance with user signal present and channel phase errors.

End of Example 5.14

Example 5.15: Use of Calibration Data to Improve the Steady-State Performance with Channel Hardware Errors

The degradation in performance with the user signal present and channel hardware errors is caused by mismatch between the steering vector and the covariance matrix (or the data used to compute the covariance matrix). Another method for improving the steady-state performance is to compensate for this mismatch by using calibration data. We assume that perfect calibration data is available so that the gain and phase errors can be precisely matched.

Calibration data can be used to modify the steering vector so that it matches the data in the covariance matrix or it can be used to modify the data in the covariance matrix so that it matches the steering vector. We consider an example of the first procedure and thus use the calibration data to modify the steering vector.

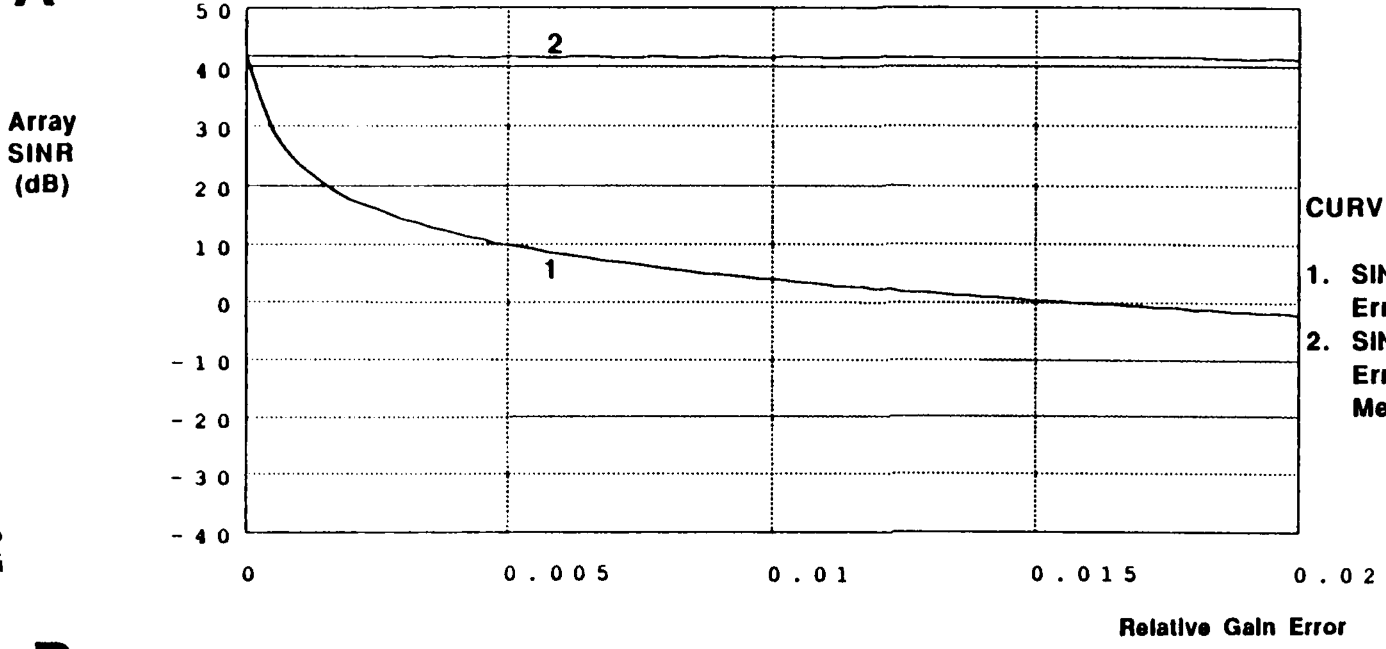
The steering vector modification technique is to multiply each element in the steering by the same gain and phase which is incurred in the corresponding hardware channel. Note that the corresponding $\tilde{\vec{u}}_0$ is equation (5.14) becomes

$$\tilde{\vec{u}}_0 = G^{-1}(G\vec{u}_0) = \vec{u}_0 \tag{5.15}$$

Curve 1 in Figure 5.20a and Figure 5.20b depicts the baseline performance of steady-

Figure 5.19: Use of Signal Subspace Algorithm to Reduce Impact of Hardware Errors on Steady-State Performance (Example 5.14)

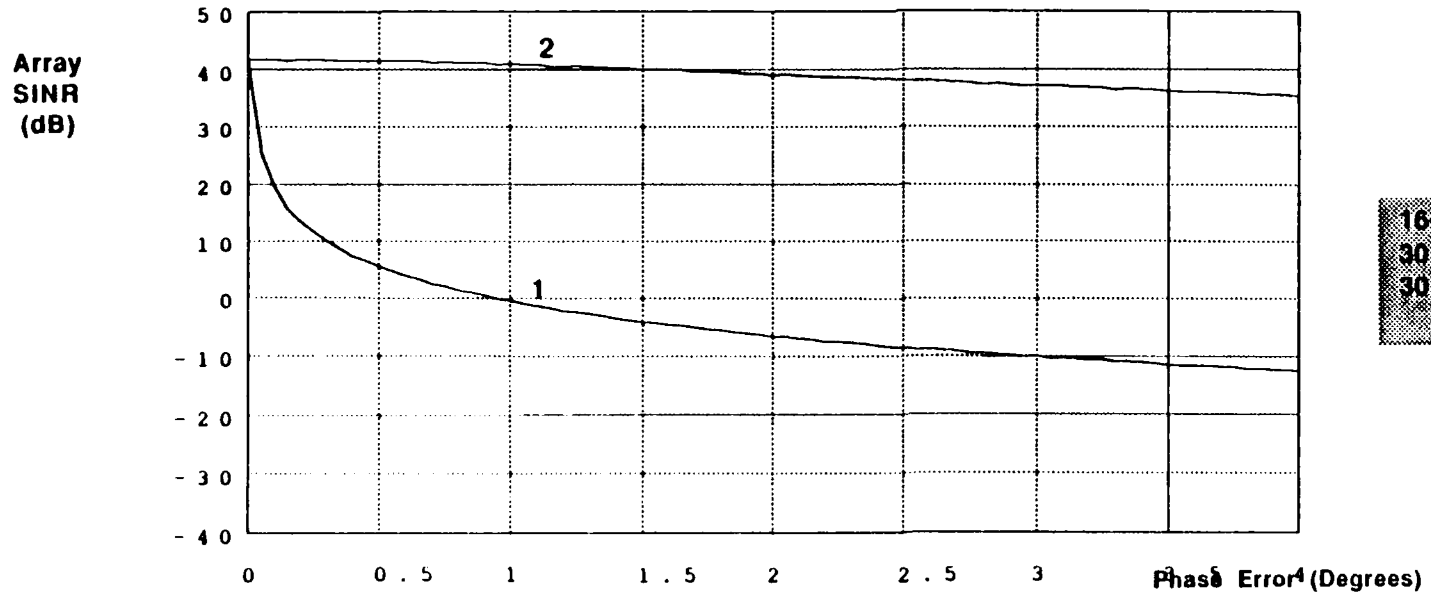
A



CURVES:

1. SINR as a Function of Hardware Errors Using SMI (Steady-State)
2. SINR as a Function of Hardware Errors Using Signal Subspace Method (Steady-State)

B



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 50 Degrees

state SMI as a function of gain and phase errors respectively. Curve 2 depicts the steady-state performance of the the steering vector modification technique based on calibration measurements. The improvement in performance is substantial.

End of Example 5.15

5.3.4 Steady-State Performance of the New Algorithms Against Hardware Channel Errors

As a consequence of removing the user signal prior to weight determination, the new class of algorithms described in Section 3.5 is comparatively ***insensitive to channel hardware errors.*** Specifically, provided that the user signal is attenuated to a level lower than the noise floor, the effects of channel hardware errors can be significantly reduced. The following example illustrates this point:

Example 5.16: Use of New Algorithm with Two-Point Pre-Filter to Reduce Impact of Channel Hardware Errors on Steady-State Performance

Consider again the scenario of Example 5.12. An example algorithm was constructed from the class described in Section 3.3 as follows:

1. The weight set $\vec{w}_{\overline{U}}$ was selected to be a two-point filter (dipole) that places a null in the assumed direction of the user (i.e., $\theta = 5°$).

2. The matrix M was taken to be the identity matrix.

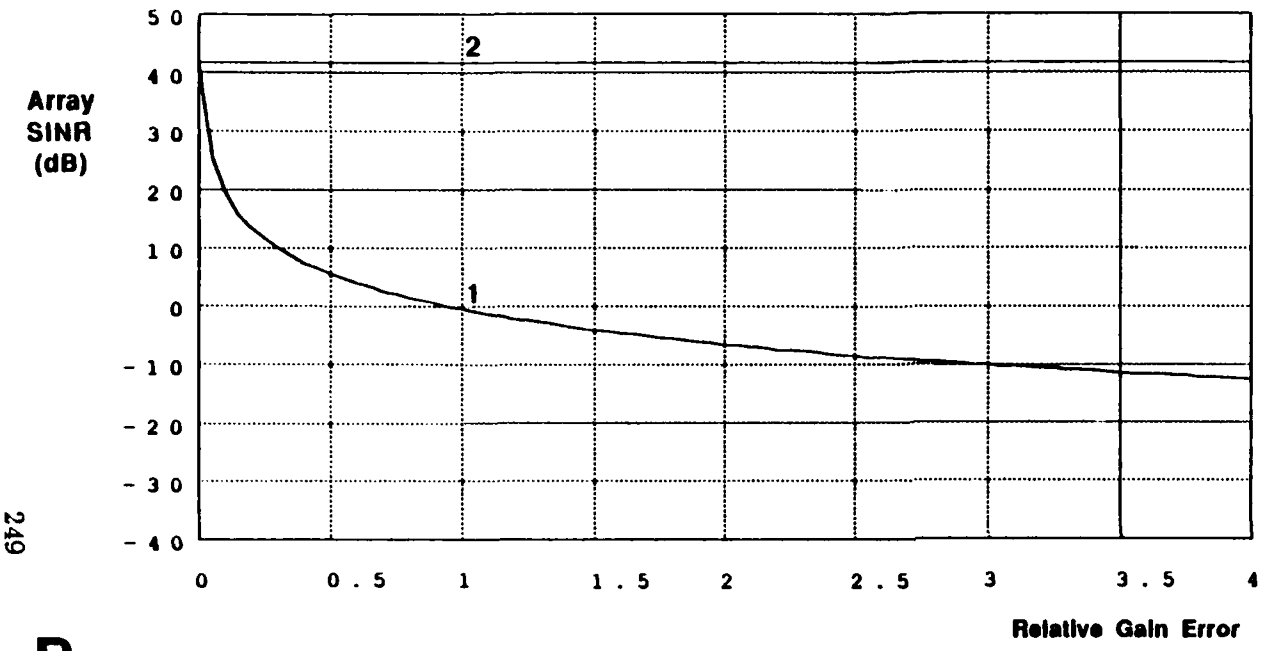
3. The weight set $\vec{w}_U$ was selected to be a two-point filter matched to the assumed direction of the user (i.e., which steered a beam in the assumed direction of the user).

4. The final weight vector $\vec{w}$ was constrained to provide unity gain in the direction of the user.

The presence of channel hardware errors causes the actual antenna pattern to depart from the constraint specified in condition 4. The sensitivity of this algorithm to the level of channel hardware errors was examined for the case of steady-state performance by using the steady-state covariance matrix R_{xx}. The steady-state SINR of the composite adaptive processor was calculated for a range of gain errors and phase errors. The results are depicted in Figure 5.21. Curve 1 in Figure 5.21a depicts the baseline performance of steady-state SMI as a function of gain errors. Curve 2 in Figure 5.21a depicts the improvement in performance

Figure 5.20: Use of Calibration Data (Steering Vector Modification) to Reduce Impact of Hardware Errors on Steady-State Performance (Example 5.15)

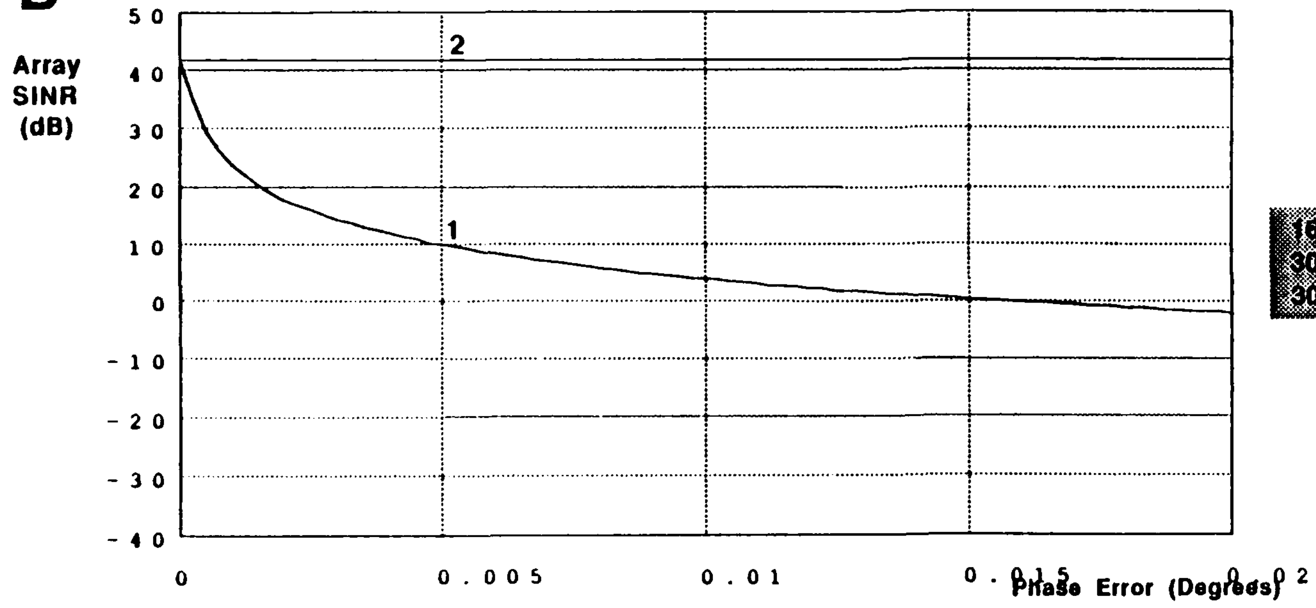
A



CURVES:

1. SINR as a Function of Hardware Errors Using SMI (Steady-State)
2. SINR as a Function of Hardware Errors Using Steering Vector Modification (Steady-State)

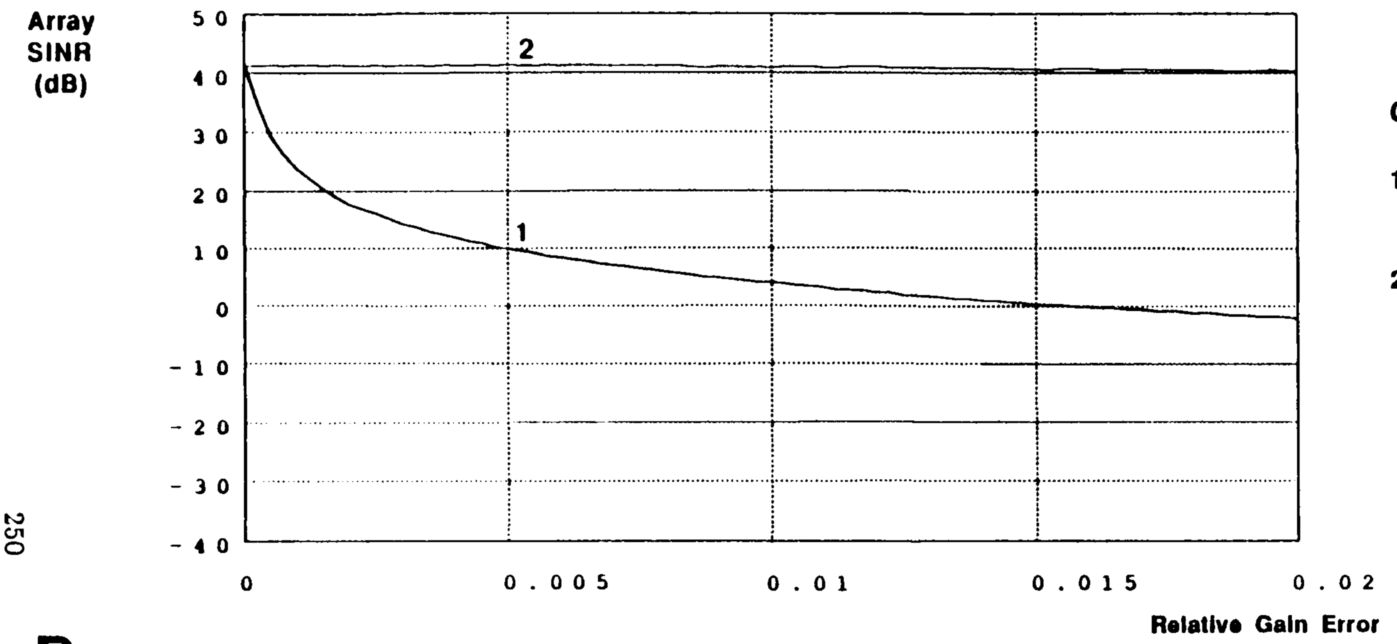
B



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 50 Degrees

Figure 5.21: Use of New Algorithm (Two-Point Pre-Filter) to Reduce Impact of Hardware Errors on Steady-State Performance (Example 5.16)

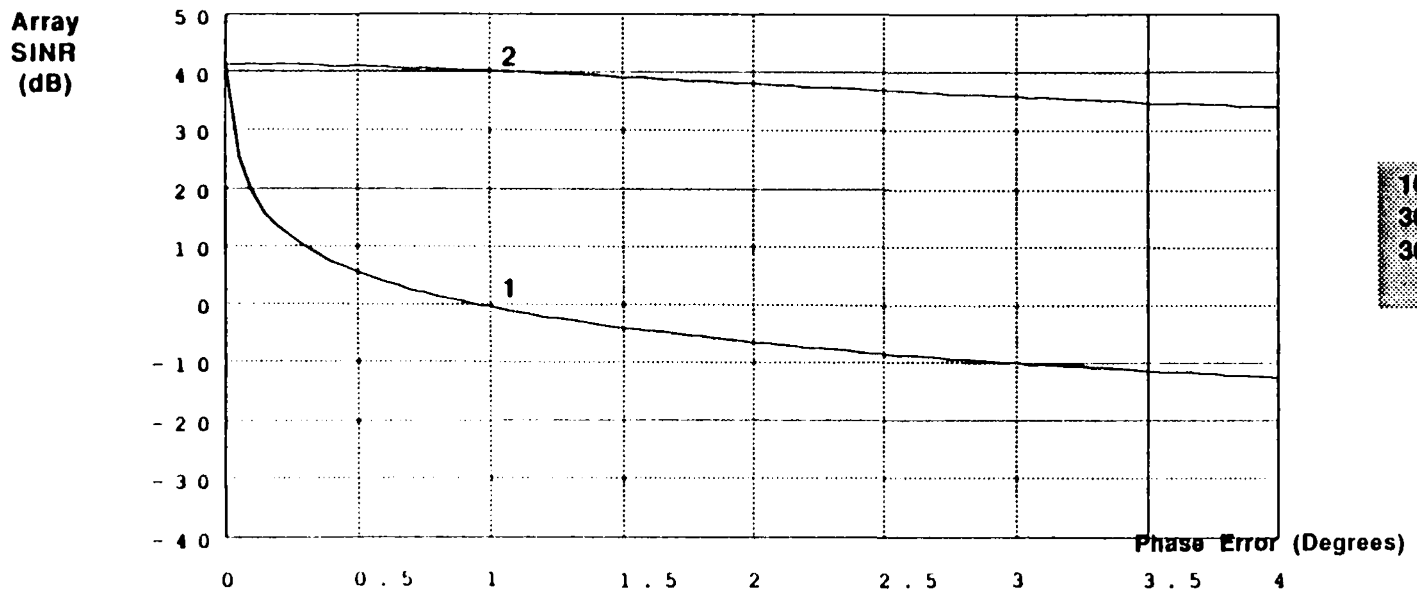
A



CURVES:

1. SINR as a Function of Hardware Errors Using SMI (Steady-State)
2. SINR as a Function of Hardware Errors Using New Algorithm with Two-Point Pre-Filter (Steady-State)

B



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 60 Degrees

attained when the two-point prefiltering algorithm is is used. The performance improvement of the new algorithm is substantial. Similarly, Curves 1 and 2 and Figure 5.21b depict the performance in the face of phase errors. The new algorithm again provides greatly improved performance.

End of Example 5.16

5.3.5 Transient Performance with Channel Hardware Errors

Whereas the methods of pseudo-noise addition, signal-subspace processing, and steering vector modification are effective in improving the steady-state nulling performance, ***these techniques are of limited utility under transient conditions.***

By contrast, the new class of algorithms described in Section 3.3 yields improved performance under ***both steady-state and transient conditions.*** The following examples illustrate these points.

Example 5.17: Conventional Algorithms with Fixes After 100 Snapshots

A sample covariance matrix $\hat{R}_{xx}$ was constructed based upon 100 snapshots. The matrix then was used in place of the (steady-state) covariance matrix R_{xx} for the conventional algorithms discussed in the prior subsection.

Figure 5.22 depicts the SINR performance as a function of pseudo-noise level (relative to the thermal noise level) using the 100 snapshot covariance matrix. From the figure, it can be seen that the optimal level of pseudo-noise yields an SINR which is ***approximately*** 25 ***dB less than the achievable SINR of*** 41.8 ***dB.*** Comparing Figure 5.22 with Figure 5.18 we see that pseudo-noise addition is far less effective in enhancing transient performance than steady-state performance.

Figures 5.23a and 5.23b illustrates the use of the signal-subspace algorithm for reducing the impact of channel hardware errors in the transient condition. Curve 1 in Figure 5.23a(b) depicts the array SINR as a function of gain(phase) errors after 100 snapshots using SMI. Curve 2 in Figure 5.23a(b) depicts the array SINR as a function of gain(phase) errors using the signal subspace algorithm after 100 snapshots. By comparing Figure 5.23 with Figure 5.19 it can be seen that the signal subspace algorithm is far less effective in enhancing transient performance than steady-state performance.

Similarly, Figures 5.24a and 5.24b illustrate the use of the steering vector modification technique for reducing the impact of channel hardware errors in the transient condition.

Figure 5.22: Addition of Pseudo-Noise to Improve Transient Performance for Fixed Set of Hardware Errors (1% Gain, 4° Phase) (Example 5.17)

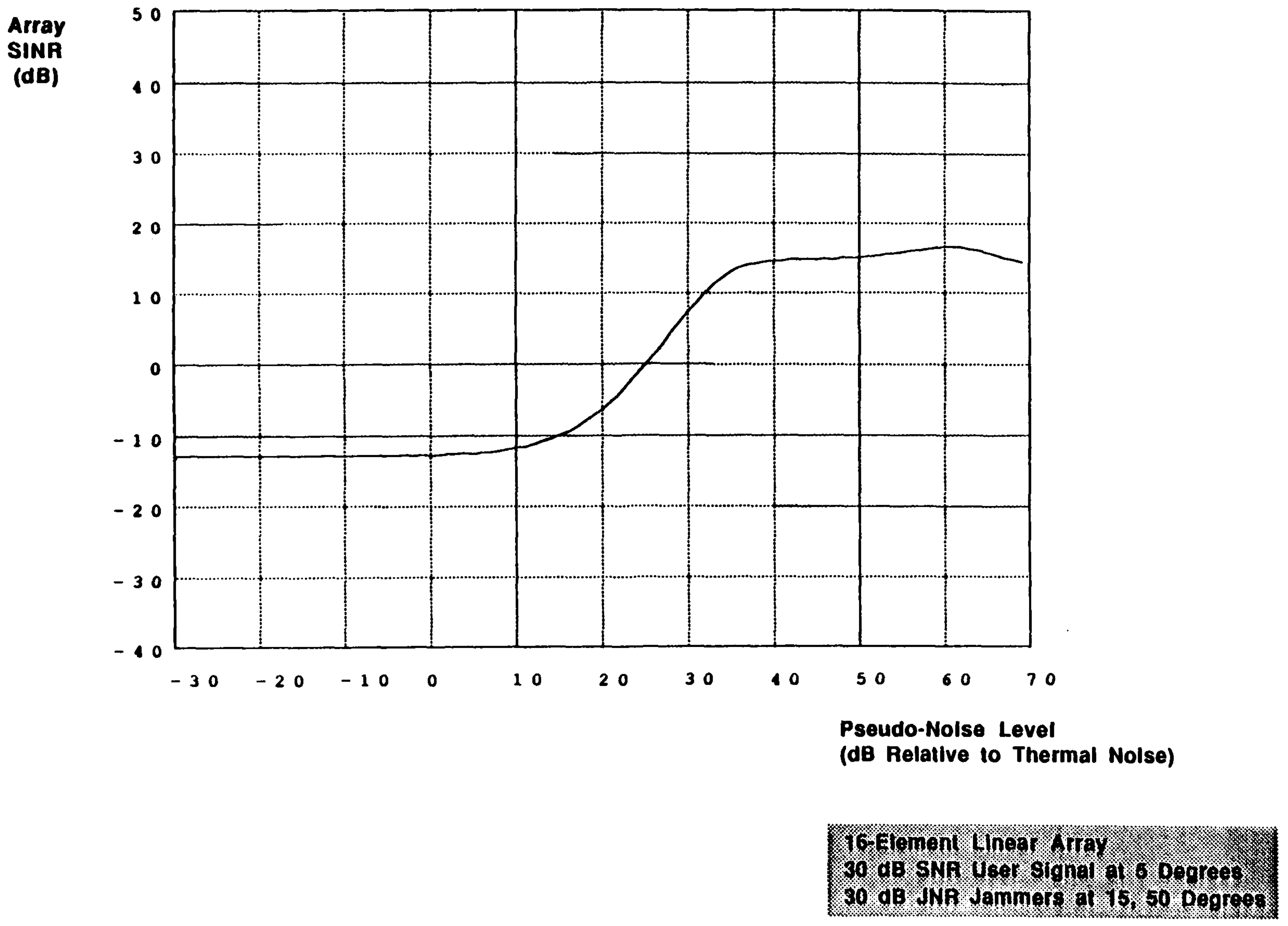
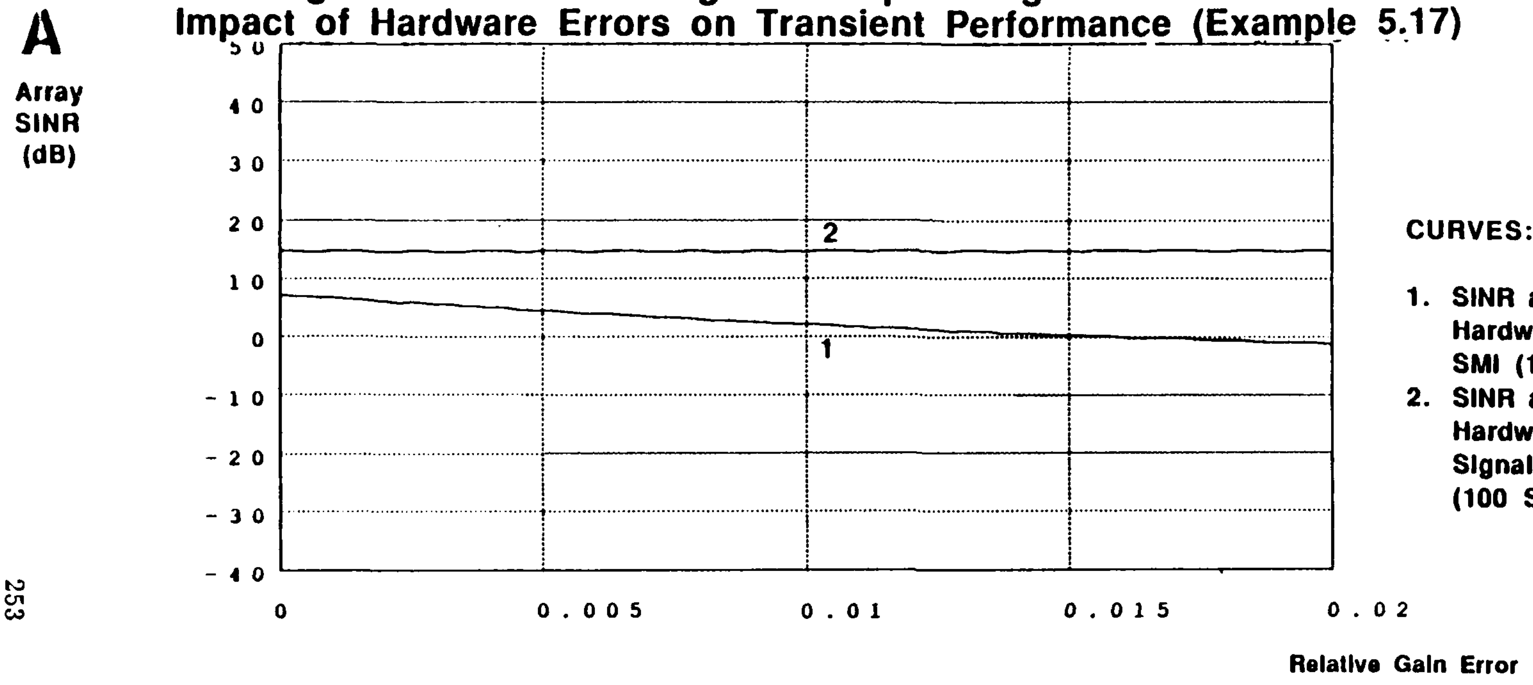
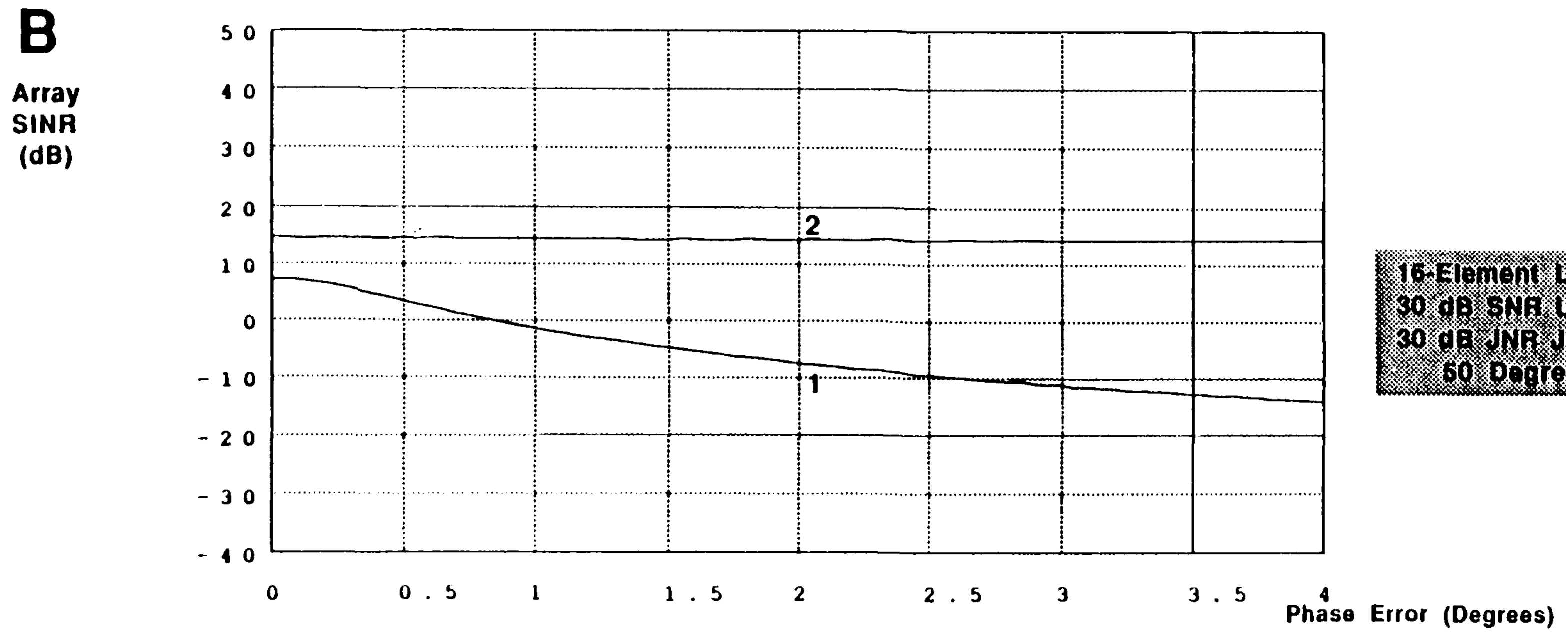


Figure 5.23: Use of Signal Subspace Algorithm to Reduce Impact of Hardware Errors on Transient Performance (Example 5.17)



CURVES:

1. SINR as a Function of Hardware Errors Using SMI (100 Snapshots)
2. SINR as a Function of Hardware Errors Using Signal Subspace Algorithm (100 Snapshots)



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 50 Degrees

Curve 1 in Figure 5.24a(b) depicts the array SINR as a function of gain(phase) errors after 100 snapshots using the steering vector modification technique described in Example 5.15. Curve 2 in Figure 5.24a(b) depicts the array SINR as a function of gain(phase) errors using the steering vector modification technique after 100 snapshots. By comparing Figure 5.24 with Figure 5.20 it can be seen that the steering vector modification algorithm which uses calibration data to match the steering vector to channel errors is far less effective in enhancing transient performance than steady-state performance.

End of Example 5.17

Collectively, Figures 5.18-5.29 and 5.22-5.24 illustrate that conventional modifications to the SMI algorithm improve transient performance when channel hardware errors are present and the user signal is present in the data used for adaptation. However, the improvement is much less dramatic in the transient condition than in the steady-state. In many cases the adaptive array provides only slight gain after 100 snapshots even though the correction is being applied.

However, the class of new algorithms described in Section 3.3 improves performance in the face of channel hardware errors in both the transient and steady-state condition.

Example 5.18 New Algorithm After 100 Snapshots

The same 100 snapshot covariance matrix $\hat{R}_{xx}$ was used to assess the performance of the new algorithm using a two-point prefilter which removes the user.

Figure 5.25 illustrates the performance of the new algorithm. Curve 1 in Figure 5.25a(b) depicts the array SINR as function of gain(phase) errors after 100 snapshots using the SMI algorithm. Curve 2 in Figure 5.25a(b) depicts the steady-state array SINR as a function of gain(phase) errors using the new algorithm. Curve 3 in Figure 5.25a(b) depicts the array SINR after 100 snapshots as a function of gain(phase) errors using the new algorithm. The performance of the new algorithm is virtually identical for the transient and steady-state conditions. Good performance is provided in both cases.

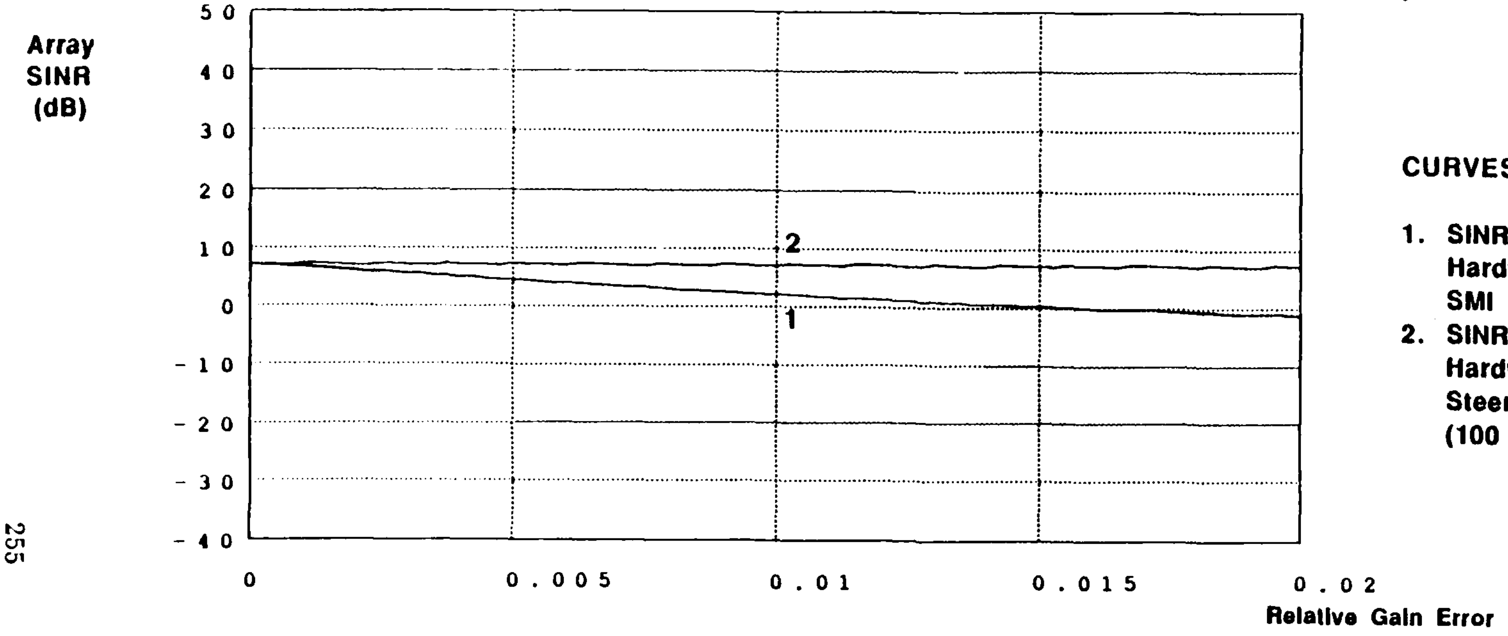
End of Example 5.18

5.3.6 Use of New Algorithm in Hybrid Architecture

In the previous subsections, we considered the use of the new algorithm within an architecture where the weights are determined and applied to the *same* of channels. This architecture might correspond, for example, to an all-digital svstem in which channel hardware

Figure 5.24: Use of Calibration Data (Steering Vector Modification) to Reduce Impact of Hardware Errors on Transient Performance (Example 5.17)

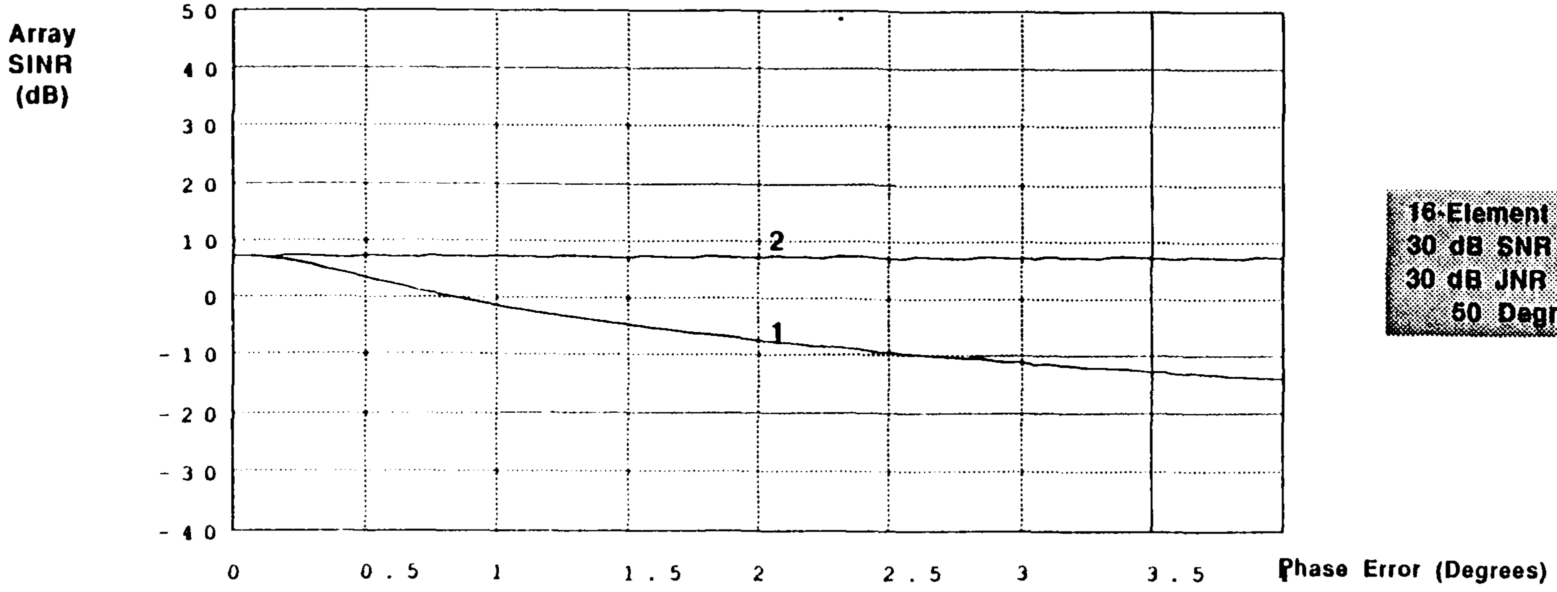
A



CURVES:

1. SINR as a Function of Hardware Errors Using SMI (100 Snapshots)
2. SINR as a Function of Hardware Errors Using Steering Vector Modification (100 Snapshots)

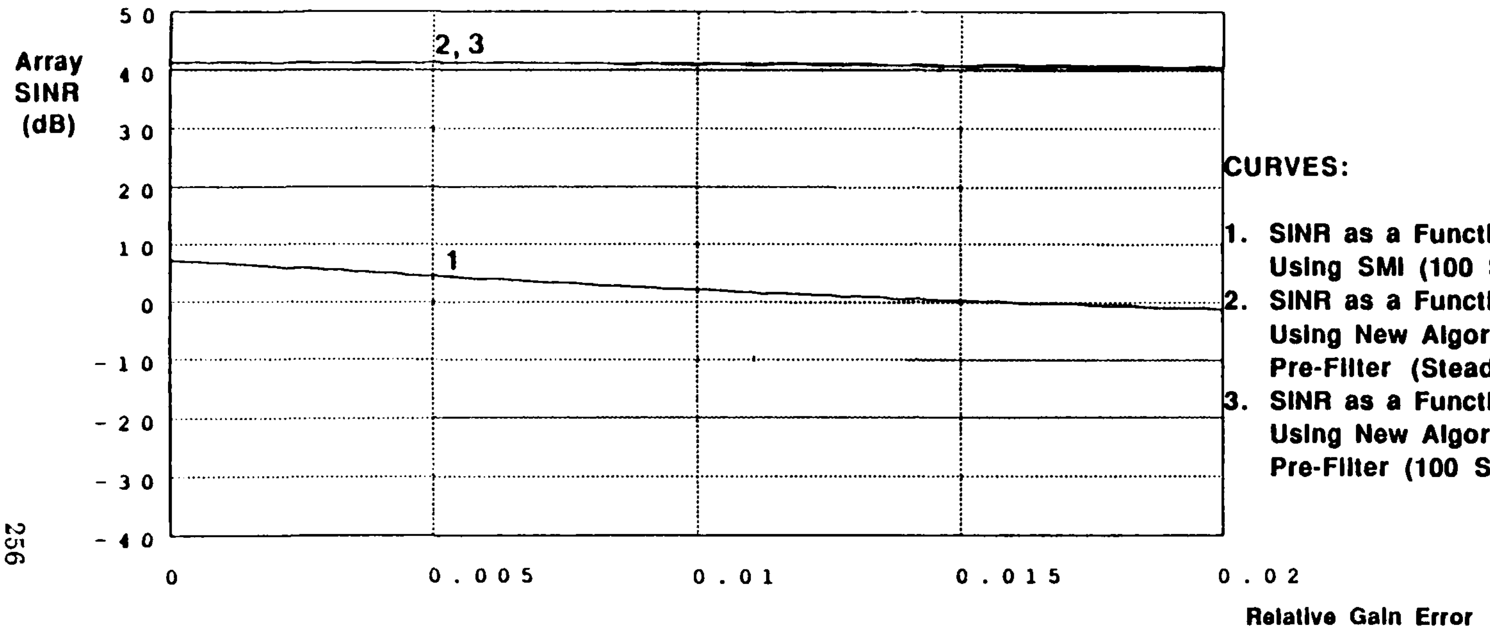
B



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 50 Degrees

Figure 5.25: Use of New Algorithm (Two-Point Pre-Filter) to Reduce Impact of Hardware Errors on Steady-State and Transient Performance (Example 5.18)

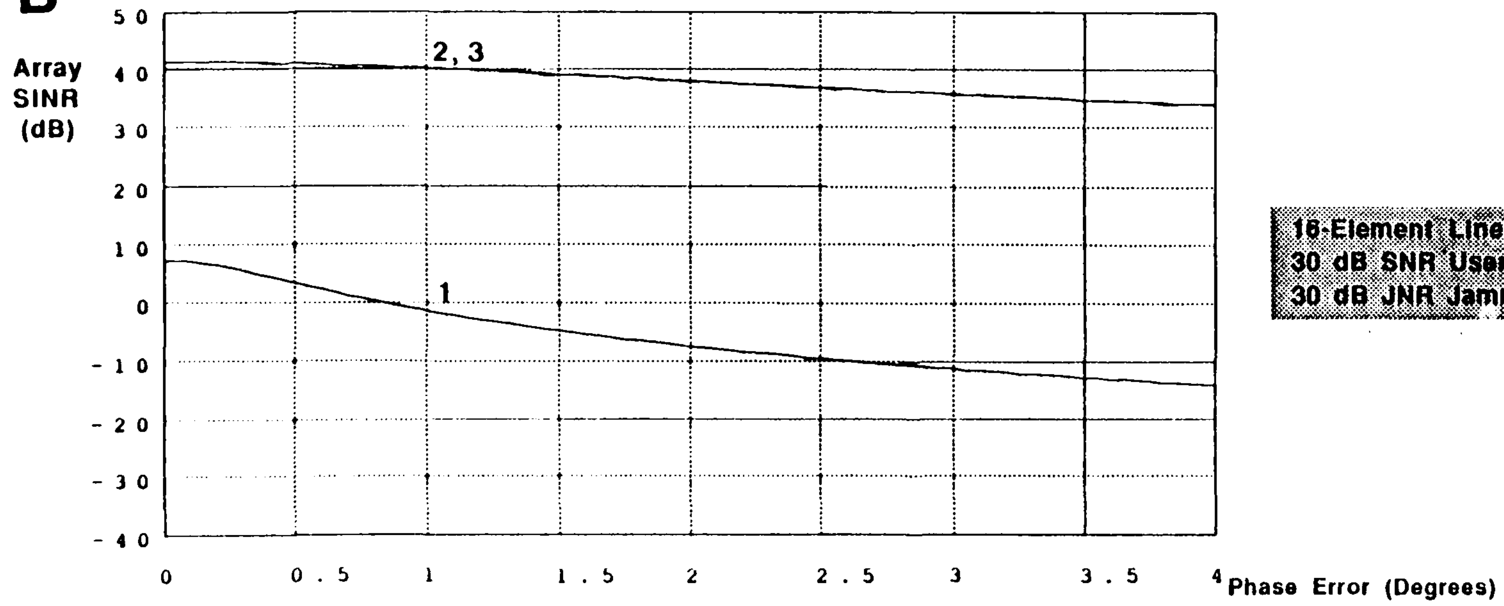
A



CURVES:

1. SINR as a Function of Hardware Errors Using SMI (100 Snapshots)
2. SINR as a Function of Hardware Errors Using New Algorithm with Two-Point Pre-Filter (Steady-State)
3. SINR as a Function of Hardware Errors Using New Algorithm with Two-Point Pre-Filter (100 Snapshots)

B



16-Element Linear Array
30 dB SNR User at 5 Degrees
30 dB JNR Jammers at 15, 50 Degrees

errors are introduced only at the front end where analog to digital conversion takes place. In this subsection we illustrate the performance of the new algorithm within a slightly different architecture. Here, we assume that that the weights are applied to a set of channels which *differ* from those used to obtain the measurement data. This architecture might correspond to a hybrid scheme in which the weights are determined from digital data and applied to a separate set of analog attenuators and phase shifters. This architecture is described in further detail in Chapter 6.

The hybrid architecture suffers loss in performance due to channel hardware errors when the user signal is present. This loss occurs whenever the user signal is present and there is a mismatch between the data used to form the covariance matrix and the assumed steering vector. In a hybrid architecture where there are separate measurement and weight application channels, an additional loss in performance can also occur. This loss is not dependent on the level of the user signal and is due to a *limit on the depth of the nulls formed.* The limitation on null depth is actually a different type of mismatch which occurs when the weights are determined from one set of data and applied against a different set of data. We illustrate this limit on achievable null depth in the following example.

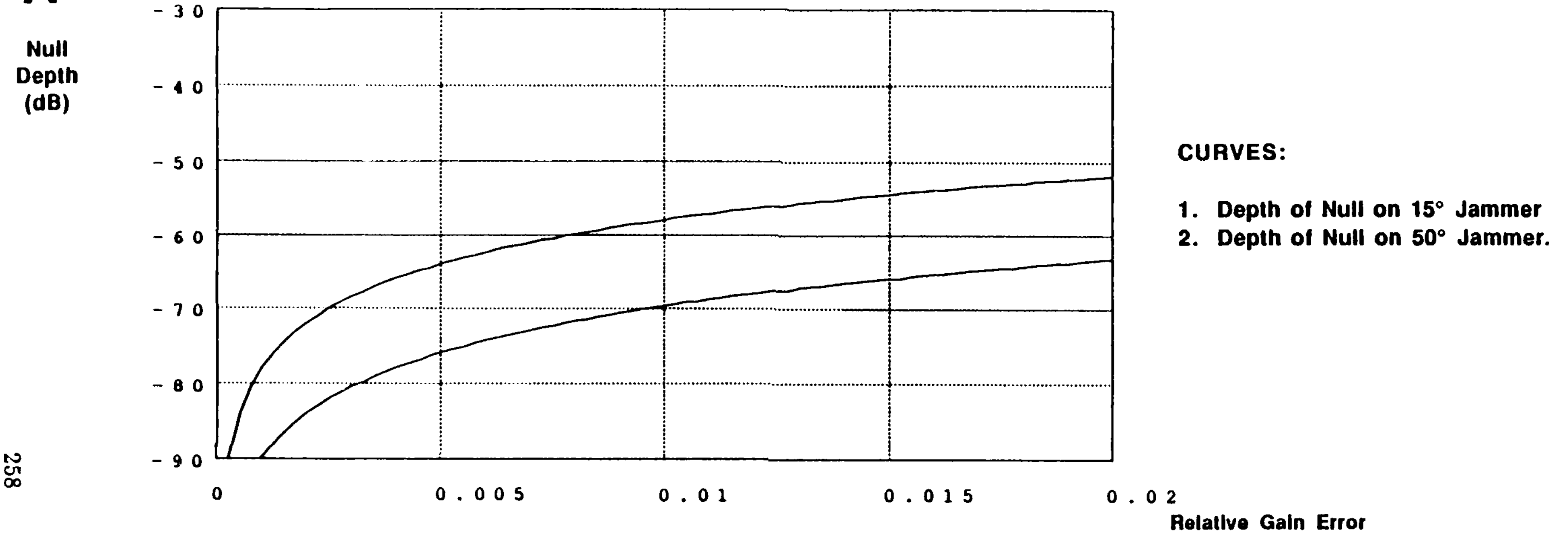
Example 5.19: Null Depths as a Function of Channel Hardware Errors when the Measurement and Application Channels Differ

We consider the foregoing scenario of a 16-element linear array in the presence of a 30 dB JNR jammer at 15° and a 30 dB JNR jammer at 50°. For this example, we assume that the user signal is not present and that the array has had an infinite amount of time to converge. The weights are determined from channels which have parametrically varying levels of gain and phase errors. However, we assume that the weights are not re-applied to the same set of channels but rather are applied to a different set of channels. For illustrative purposes, we assume that the weight application channels are error-free. The limitation on null depths are due to the fact that the weights are determined from a set of channels with errors but are applied to a set of error-free channels.

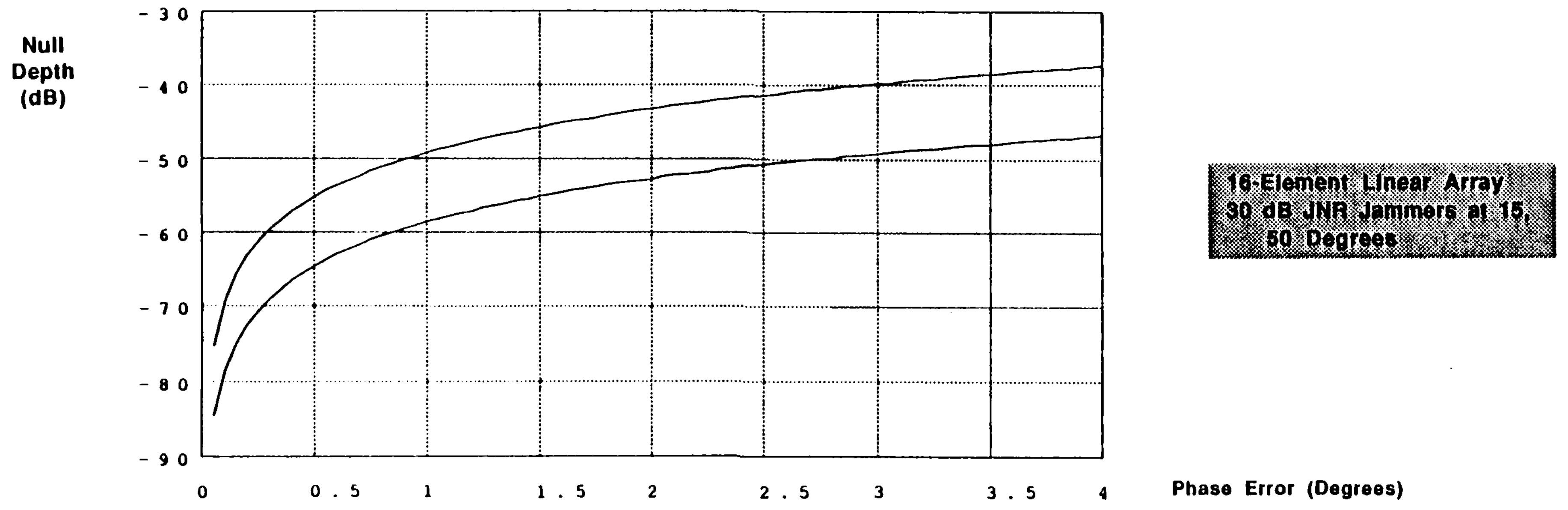
Figure 5.26 depicts the performance of the hybrid nulling architecture as a function of the level of hardware errors. The curves indicate the depth of the null on the 15° jammer (Curve 1) and the 50° jammer (Curve 2) as a function of gain error (Figure 5.26a) and phase error (Figure 5.26b). The null depth is measured relative to the unit gain constraint in the user direction of 5° i.e. the depth of the null in the 5° look direction is 0 dB. Note that the depth of the nulls on both jammers decreases as the level of relative error between

Figure 5.26: Null Depths as a Function of Channel Hardware Errors for Hybrid Architecture (No User Signal Present)

A



B



the weight determination and weight application channel increases. The null is deeper on the 50° jammer since it is located further out in the sidelobes on the antenna pattern. Note that this effect is independent of the user level and will ultimately limit the performance in high JNR scenarios.

End of Example 5.19

The new algorithms presented in Section 3.3 can also be applied to the hybrid nulling architecture. The resulting system exhibits rapid convergence, insensitivity to pointing errors, and improved tolerance to hardware errors. The algorithm does not compensate for the limitation on achievable null depth described in Example 5.19. Rather, it limits the undesirable effects caused by the user signal being nulled in a hybrid architecure. The following example illustrates the improvement in performance.

Example 5.20: Use of the New Algorithm in a Hybrid Architecure

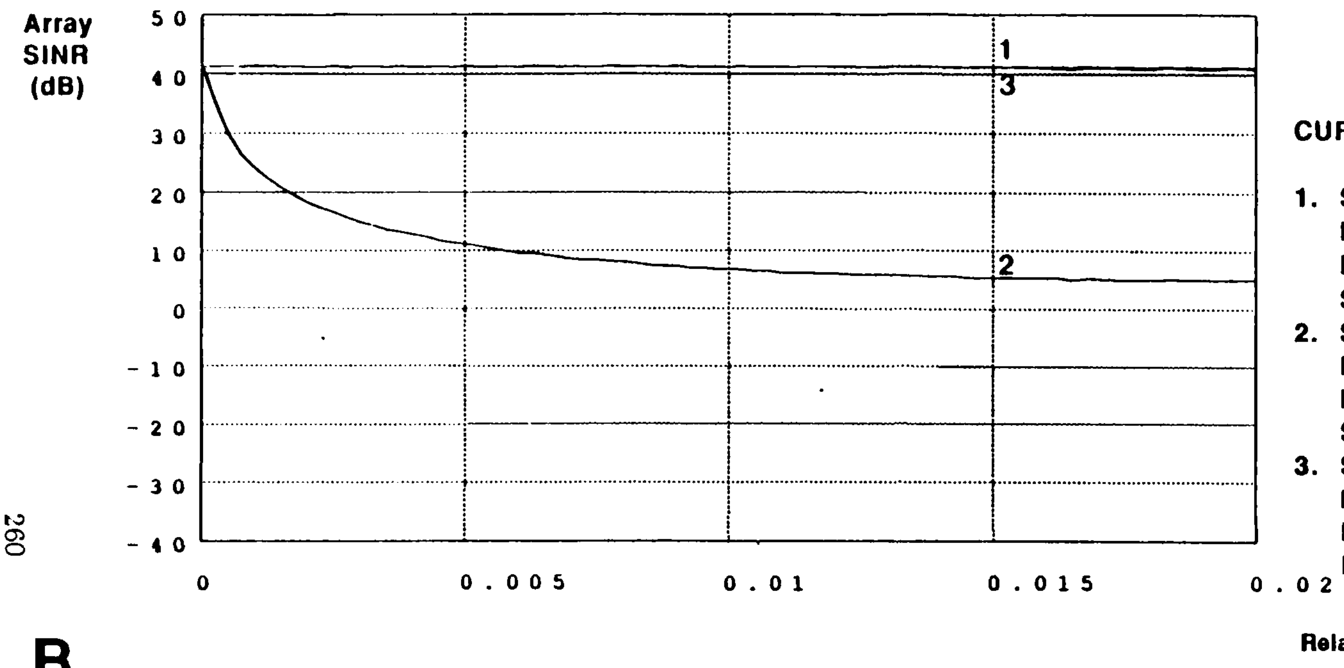
We consider the same scenario as in Example 5.19 except that the user signal is present at 30 dB SNR. For the range of gain errors and phase errors we selected, the limit on null depth does not substantially impact performance, i.e. the array provides 0 dB relative gain to the 30 dB user while providing -38 dB gain to the JNR 15° jammer when 4° phase errors are present. However, the mismatch between the steering vector and the actual user data in the covariance matrix again causes a serious loss in performance unless compensated for.

Figure 5.27 illustrates the performance of the hybrid architecture as a function of the level of gain errors (Figure 5.27a) and phase errors (Figure 5.27b). Curve 1 depicts the array SINR as a function of hardware error when the user signal is not present in the data used for adaptation. Note that the SINR drops only slightly over the entire range of hardware errors. This slight drop is attributed to the limitation on achievable null depth described in Example 5.19. Curve 2 depicts the the array SINR when the 30 dB user signal is present in the data used for adaptation. Note that the SINR decreases rapidly as the level of the error increases. Finally, Curve 3 depicts the array SINR when the 30 dB user signal is present in the data used for adaptation and the new algorithm using a two-point pre-filter is used. The new algorithm exhibits a dramatic improvement in performance when the weights are determined from and applied to separate sets of channels.

End of Example 5.20

Figure 5.27: Use of New Algorithm (Two-Point Pre-Filter) to Reduce Effects of Hardware Errors for Hybrid Architecture

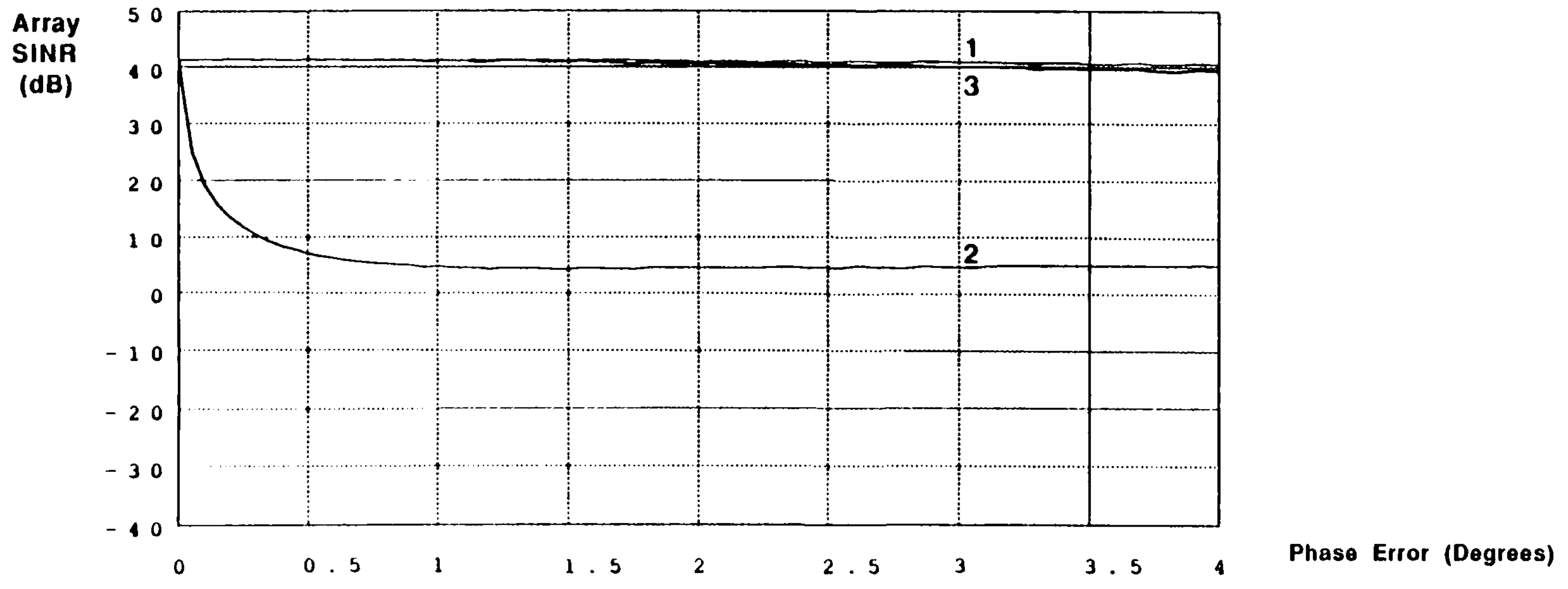
A



CURVES:

1. SINR as a Function of Hardware Error When User Signal Is Not Present In Adaptation Data. Steady-State SMI.
2. SINR as a Function of Hardware Error When 30 dB SNR User Signal Is Present In the Adaptation Data. Steady-State SMI.
3. SINR as a Function of Hardware Error When 30 dB SNR User Signal Is Present In the Adaptation Data. New Algorithm In Steady-State.

B



Bibliography

[1] J.E. Hudson, *Introduction to Adaptive Arrays*, Chap. 5, Peter Peregrinus Ltd, New York and London, 1981.

[2] S. Applebaum and D. Chapman, "Adaptive Arrays with Mainbeam Constraints", *IEEE Trans. Antennas and Propagat.*, Vol. AP-24, No. 5, pp. 650-661, Sept 1976.

[3] J. Maksym, "A robust formulation of an optimum cross-spectral beamformer for line arrays", *J. Acoust. Soc. Am.*, Vol. 65, pp.971-975, 1979.

[4] J. Griffiths and J. Hudson, "An introduction to adaptive processing in a passive sonar system", in *Aspects of Signal Processing*, Proceedings of the NATO Advanced Study on signal processing and underwater acoustics, Porto Venere, Italy, 1976, (Reidel, 1977).

[5] W.F. Gabriel, "Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems", *IEEE Trans. Antennas and Propagat.*, Vol. AP-34, No. 3, pp. 291-300, Mar. 1986.

[6] W. White, "Artificial Noise in Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-14, No. 2, pp. 380-387, Mar. 1978.

[7] M. Er and A. Cantoni, "Derivative Constraints for Broadband Element Space Antenna Array Processors", *IEEE Trans. Acoust., Speech, Sig Proc.*, Vol. ASSP-31, pp. 1378-1393, Dec. 1983.

[8] H. Cox, "Resolving Power and Sensitivity to Mismatch of Optimum Array Processors", *J. Acoust. Soc. Am.*, Vol. 54, No. 3, pp. 771-785, Sept. 1973.

[9] A.M. Vural, "Effects of Perturbations on the Performance of Optimum/Adaptive Arrays", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-15, No. 1, pp. 76-87, January 1979.

[10] J. Mayhan, "Bandwidth Limitations on Achievable Cancellation for Adaptive Nulling Systems", Lincoln Laboratory Technical Note 1978-1, Feb. 1978.

Chapter 6

Example System Architecture

Contents

List of Figures

6.1 Introduction

This chapter describes an example architecture for a satellite on-board nulling system for wideband signals. The main conclusion is that digital implementation of controllers for such systems may now be practical in a variety of applications.

6.2 System Architecture

Figure 6.1 depicts the candidate hardware system architecture. The architecture, which is applicable to both phased array and multi-beam antenna systems, is specifically oriented toward the EHF satellite communications band. However it can be easily modified for use at other frequencies. The design seeks to minimize hardware size, weight, complexity and power consumption in order to be compatible with satellite on-board deployment requirements.

Salient features of the architecture include:

1. Beamforming weights $w_1 \cdot \cdot w_K$ are applied to individual sensor (element) signals directly at RF, using digitally controlled attenuators and phase shifters. The beamformed output is also developed at RF.

2. To facilitate weight determination, array snapshots are produced by simultaneously sampling down-converted versions of the element signals using a bank of sub-nanosecond sample-and-hold devices. Down-conversion is to baseband and sampling is effected in both in-phase and quadrature baseband channels. Bandwidths of the I and Q baseband signals are 1 GHz. The output of each sample-and-hold element is digitized to 8 bits.

3. The beamforming weights are determined numerically from the snapshot data in a flexible digital computational module. The digital unit can support a broad range of nulling algorithms. Outputs of the computational subsystem are digital commands to the attenuators and phase shifters.

4. The computational subsystem also receives a (complex) sample of the beamformed output signal concurrently with each array snapshot. This quantity is used for calibration purposes and also provides a mechanism for optional closed-loop system operation. Although algorithm work in this program primarily focused on open-loop

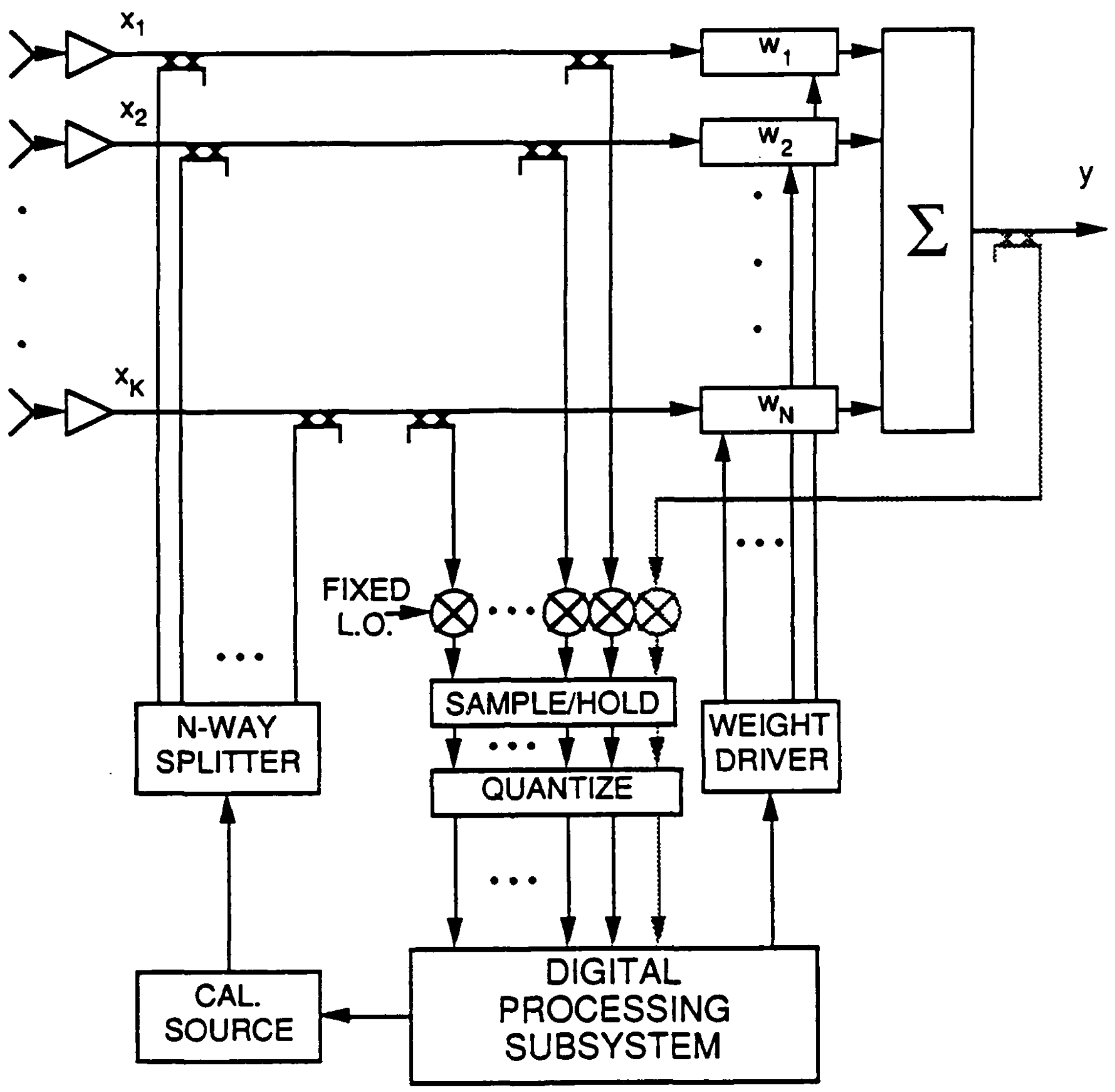


Figure 6.1: Candidate Nulling Architecture

nulling methods, the incorporation of a closed-loop option into the architecture allows for possible two-tier operation. In this approach the open-loop calculations are used for achieving rapid, albeit sub-optimum, weight sets which serve as initial states for subsequent closed-loop refinement.

The following sections summarize the design rationale, cost/performance tradeoffs, and hardware component options for the overall systems architecture and for the major subsystems shown in Figure 6.1.

6.3 Design Rationale

Major drivers in selection of the architecture include desire to achieve effective nulling over a 2 GHz instantaneous RF bandwidth, a requirement to determine the adaptive beamforming weights numerically (i.e., via digital computer) in order to support the broadest possible set of advanced algorithms, and a need to minimize the amount of hardware (i.e., size, weight, power) in order that the design be compatible with satellite implementation constraints.

Operation over an instantaneous 2 GHz band precludes the use of digital beamforming techniques, since this approach would require massively parallel computing structures for real-time application of the nulling weights, and either extremely fast (2 GHz), high precision ($\geq$ 8 bits) A/D converters (beyond the capability of current technology) or a complex channelization approach that would employ vast numbers of slower converters. Solutions requiring massive amounts of hardware are impractical from a satellite implementation point of view. Accordingly, we have opted to effect the beamforming via analog methods, using digitally controlled attenuators and phase shifters for application of the beamforming weights.

Although beamforming weights will be applied at RF, the calculation of these weights needs to be effected in a flexible digital processor in order to accommodate open-loop adaptive algorithms such as Sample Matrix Inversion and closed-loop techniques like Applebaum-Howells, and to implement the rapidly convergent and broadband/robust nulling methods described in Chapters 2-5. A characteristic of the weight estimation algorithms developed in Chapters 2-5 is that they can operate on snapshotted array data that is collected at a much slower rate than what would be required for Nyquist sampling of the element signals. However, the snapshot data should reflect the full signal bandwidth. This translates to a need for large-scale channelization or for sub-nanosecond sample-and-hold circuits, followed by A/D converters that can operate at a much slower snapshotting rate. Recently developed sample-and-hold devices are capable of supporting the 2 GHz bandwidth requirement, and use of these devices avoids the hardware duplication that characterizes the channelization approach.

A fundamental property of this system architecture is that the beamforming weights are applied in a physical path that is different from the one used for making measurements. This leads to the potential for substantial differences between the desired beam pattern and what is actually achieved. The problem can be mitigated by careful calibration of the path differences, using on-board calibration resources. In the proposed architecture, calibration software resides in the same digital processing module that calculates the beamforming weights. We envision the calibration process to be a background task, operating during system "slack" times. A sample of the beamformed output signal is included in the set of inputs to the digital processor in order to support the calibration process. The same data is useful for supporting closed-loop operation nulling algorithms (e.g., Applebaum-Howells).

6.4 Subsystem Descriptions

6.4.1 Beamforming Network

The weights are applied at EHF using phase shifters and variable power dividers. This technique is generally preferable to schemes in which the signal from each element is downconverted (typically to C band) and then weighted using PIN diode-based vector modulators, the larger fractional bandwidth resulting from downconversion offsetting any potential gains from operation at lower frequency. This approach is quite conventional and has been used on all DSCS satellites to date. This technology is well-established and many systems of this type have been built by Electromagnetic Sciences (EMS) of Atlanta, GA, including a 16 element nulling system built for TRW and a 4 element system for MIT Lincoln Laboratory. EMS has supplied both 19 and 61 element beamforming systems for previous DSCS satellites and has been involved in several teams' proposals for EHF add-ons to DSCS.

The essential component of the weighting systems is the ferrite phase shifter, a device whose phase shift can be precisely set by changing the magnetization of the ferrite in a controlled manner. Typically this is accomplished by first applying a large magnetizing pulse to the ferrite to drive it to saturation and then applying pulses of opposite polarity and of tightly controlled amplitude so as to reduce the magnetization of the ferrite to the value appropriate for the desired phase shift. The relationship between the amplitude of the applied current pulse and the resultant phase shift is nearly linear and can readily be linearized (calibrated) so that 8 to 10 bits of resolution over a span of 360° can be achieved. The variation of phase shift over a 2 GHz band at 44 GHz is typically $\leq 1°$ rms and is comparable to one LSB of resolution. Two phase shifters, a 90° hybrid, and a 180° hybrid can be combined to form a variable power divider (VPD).

The phase shifters provide linear phase quantization, i.e., the phase is adjusted in steps of about 1°. The VPD's provide amplitude weighting proportional to the sine or cosine of linearly quantized phase; thus the amplitude resolution is very small when the amplitude weight is near unity and is coarser when the amplitude weight is near zero. The amplitude quantization is not likely to limit performance in phased array applications where all the amplitude weights are near full scale (at least against a single large jammer).

Based on discussions with EMS, a beam forming network (BFN) of this type, suitable for spacecraft applications, and sized to handle 37 elements would weigh approximately 15 pounds and would consume about 9 Watts in a static mode, i.e., while the beamforming weights are not being changed. Changing state of either a phase shifter or a VPD requires about 20 ujoules; changing the entire weight set requires approximately 1.5 mjoules. As an example, changing the weight set at a 1 KHz rate would require about 1.5 Watts. These weight and power estimates, which include the necessary weight driver circuits, are based on BFNs for MBA applications where the size of the MBA itself and the need to locate the BFN immediately behind the MBA largely dictate the size of the BFN. For a phased array, the size, and consequently the weight, might be slightly different.

6.4.2 Correlation Subsystem

In order to determine the required weights for the Beam Forming Network (BFN), it is necessary to cross-correlate various pairs of signals. There are several methods to accomplish this, each with its own advantages. Two distinct methods will be illustrated here.

Figures 6.2 and 6.3 depict two possible approaches to cross-correlating signals. The salient characteristic of both configurations is the use of a single time-shared correlator to perform all cross-correlations. This approach has the advantage of minimizing the number of correlators, and correlating all pairs of signals with a common device. The approach has the disadvantages of cross-correlating different pairs of signals at different times, thereby producing perturbed correlation vectors. As a consequence, the resultant nulling weights

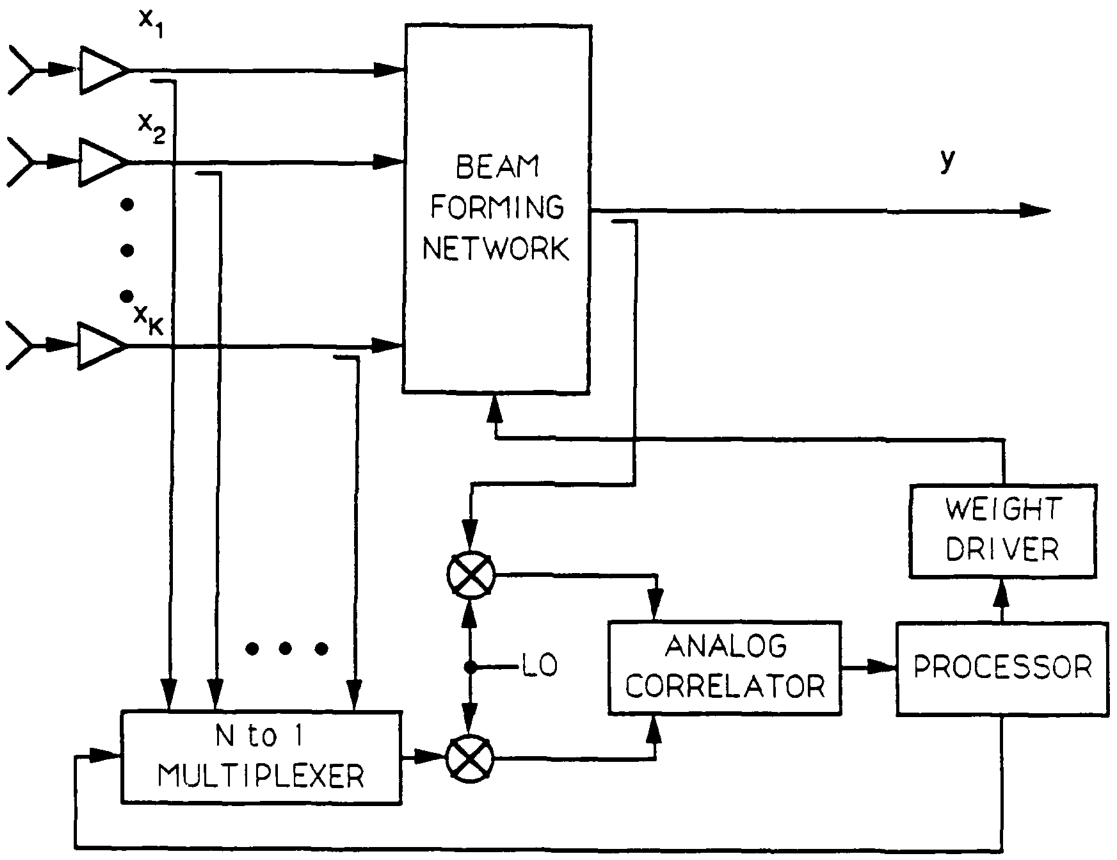


Figure 6.2: Conventional Nulling Loop

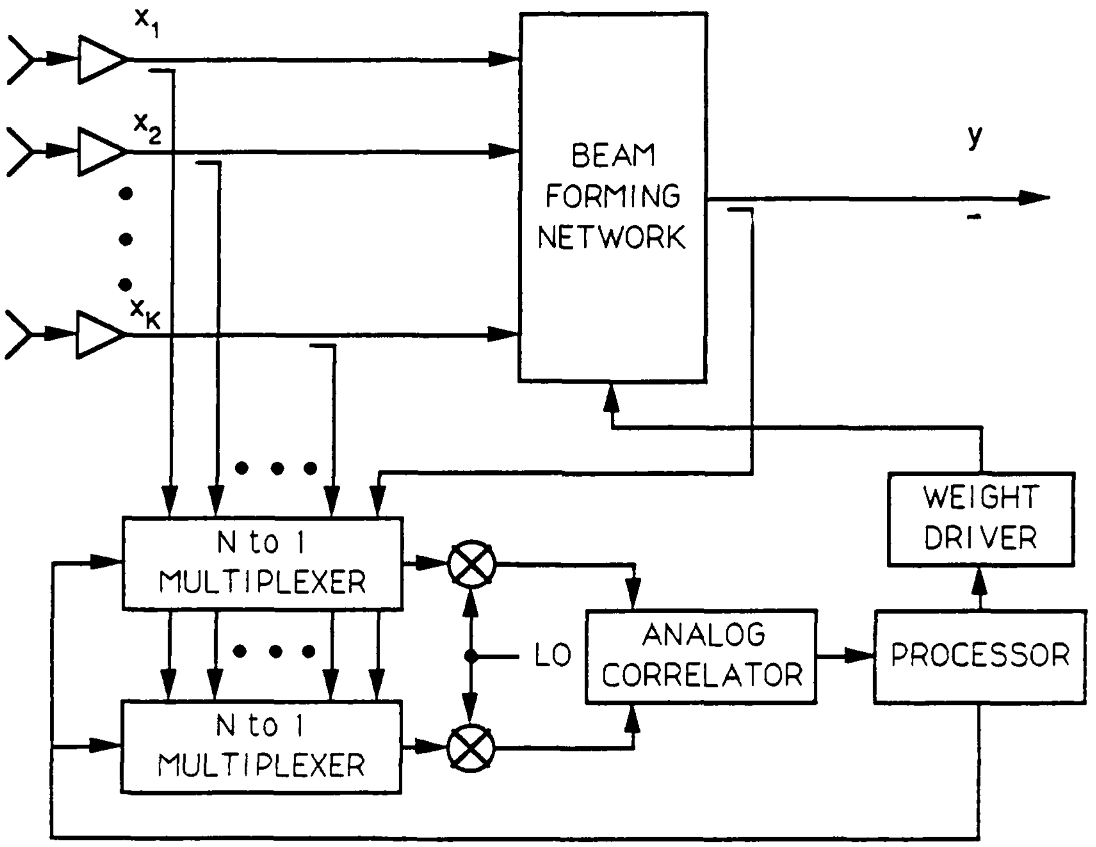


Figure 6.3: Modified Conventional Hardware

are suboptimum. Nonetheless, the weights can be quite effective in slowly varying electromagnetic environments.

Figure 6.2 illustrates the Conventional Nulling Loop. This system implements the Applebaum/Howells algorithm, i.e., it adjusts the BFN weights until the formed output is uncorrelated with each of the element signals. Since it is not practical with current technology to perform correlations at EHF, both the input signals and the BFN are downconverted to a lower frequency region (typically 6 GHz) where analog correlators with 2-4 GHz input bandwidths are easily realized. These correlators consist of 0° and 90° hybrids and square law detectors arranged to form a quarter-square multiplier. Downconversion from EHF was previously considered too costly in terms of size and weight to be provided for every element, so the technique of multiplexing, i.e., selecting just one input at a time to be downconverted for cross-correlation, was developed (see Reference [5]). Multiplexers based on switched ferrite circulators are a mature technology and have been supplied by ElectroMagnetic Sciences (EMS) as an integral part of the BFNs described in Section 6.2. A multiplexer for 37 inputs would weigh about 1 pound and consume about 0.5 Watt while stepping channels every 1 msec, and thus it is well-suited for spacecraft applications. These BFN and Multiplexers systems from EMS also include a Cross-Guide Coupler which connects the element signals to the BFN as well as coupling a small fraction (10 dB) of the input signal to the multiplexer. The cross-guide coupler has the important feature that all the coupled input-to-multiplexer paths are phase-matched, as well as the direct input-to-BFN paths.

In operation, the processor sequentially sets the multiplexer to select each of the coupled input signals to be downconverted and correlated with the (similarly downconverted) output of the BFN. The complex output of the analog correlator is quantized by the processor. The correction to each weight w_i is proportional to the correlation of input i and the BFN output.

This architecture has been attractive for space applications because it requires a minimum of hardware. The hardware minimization has been achieved at the expense of adaptation time, since only one correlation at a time can be performed (the classical Applebaum-Howells loop correlates all N inputs simultaneously and hence, adapts N times as fast). In many applications, this longer adaptation time is quite acceptable.

Figure 6.3 is a more general configuration utilizing a single correlator. Whereas the correlations in Figure 6.2 are performed only between the array output y and selected sensors outputs $x_1, \cdots x_K$, the correlations in Figure 6.3 also can be performed between pairs of sensor outputs x_i and x_j facilitating construction of covariance matrices in addition to correlation vectors. As a result, a much broader class of algorithms can be implemented using this configuration.

Given the wide bandwidth of the signals being correlated, it is imperative for both of the configurations shown in Figure 6.2 and 6.3 that the time delays of all channels be matched to a small fraction of a nanosecond. This requires that the path delays from each element to the correlator be matched. Matching the delays to 10 psec implies mechanical tolerances on the order of 0.1 inch which is readily achievable with available hardware previously described.

Figure 6.4 depicts an alternative correlation technique in which correlations are performed digitally. The sensor signals are first downconverted to quadrature baseband. Then, the sensor signals are sampled by sample-and-hold devices having a sub-nanosecond aperture An example of a recent device exhibiting a subnanosecond time is the Tektronix TKAD20C. Next the samples are digitized at a much slower rate [e.g. 20 MHz] to produce in-phase and quadrature samples. The samples are then pipelined and passed through one or more high-speed multipliers [e.g. the Plessey 20 MHz 16 × 12 complex number multiplier, part number PDSP16112]. Thus, increments $x_i x_j$ to the elements

$$(\hat{R}_{xx})_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_i x_j^* \tag{6.1}$$

of the correlation coefficient $(\hat{R}_{xx})_{ij}$ are generated at a rate of $M \times 20$ MHz for M multipliers.

Alternately, multipliers can be used to first project the sensor samples x_j into an appropriate thinned beamspace as described in Section 2.5; then the same or other multipliers can be used to generate appropriate beamspace correlation coefficients to facilitate efficient beamspace processing. Thus, the configuration has the flexibility to produce the correlation coefficients required by most nulling algorithms.

Since the sampling times for all of the sensors are synchronized, all of the correlation coefficients apply to the same time epoch. This contrasts with the techniques of Figures 6.2 and 6.3 in which different correlation coefficients apply to different epochs. Thus the potential exists for more correlation coefficients which are more nearly coincident and hence yield more accurate nulling weights in dynamic environments. However, it must be noted that this architecture requires more attention to minimizing sample clock skew between elements. This careful time-alignment is not required simply to keep all correlations in

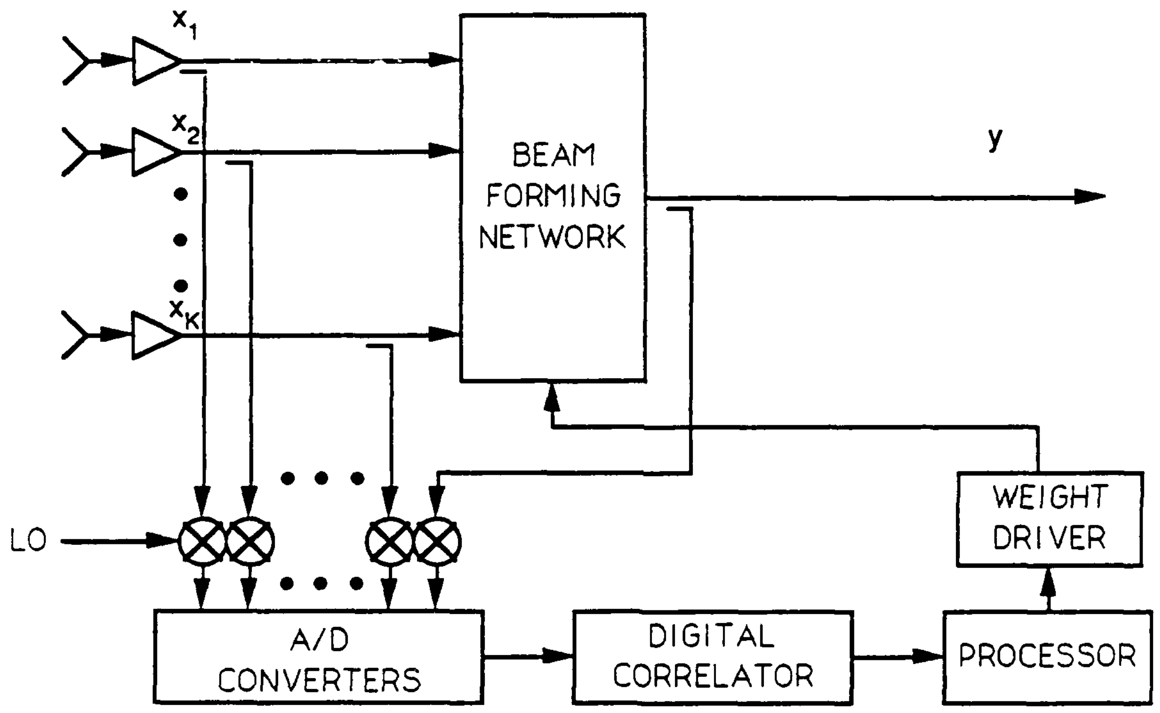


Figure 6.4: Candidate Hardware Approach

the same time epoch but rather to maintain the accuracy of the correlators over a 2 GHz bandwidth, even in a static environment.

The I and Q sample from a single channel must be taken synchronously in order to obtain an accurate representation (i.e. without image distortion). In addition, for any pair of channels, it is necessary to minimize the sample clock skew, τ, between channels so that the cross-correlation, $r_{ij}(\tau)$, is evaluated for $\tau = 0$. For a 2 GHz bandwidth, the maximum allowable τ must be a small fraction of a nanosecond. It is not practical to hold clock skew to better than 100 psec by controlling cable lengths and selecting component delays. To achieve the 10 psec or better clock skews for this application, an active clock skew adjustment circuit will be required. An example of such a circuit is the Analog Devices 9500 Programmable Delay Generator which can vary the clock delay with 10 psec resolution.

Note that stringent channel matching requirements are imposed upon all techniques as a result of the wide bandwidths of the signals. One possible method for relaxing these requirements is to utilize the multiple-null techniques described in Section 4.2. These techniques have the useful property of providing jammer suppression across a frequency band that extends well beyond the actual processing band. Thus, for example, the correlations could be performed on signal components in a 100 MHz sub-band at the center of the 2 Ghz band of interest, and nulling weights could then be calculated upon the basis of the sub-band correlation data. The multiple nulls then would extend the interval of effective jammer suppression to a significantly wider band.

6.4.3 Digital Processing Subsystem

The requirements for the Digital Processing Subsystem depend strongly upon the scenario. However, by making appropriate use of the techniques described in Chapters 3-5, and referring to commercially available processing modules, it is apparent that reasonable performance can be achieved in a variety of applications.

To provide an estimate for sizing purposes, we have chosen the following scenario, algorithm, and implementation assumptions:

1. The sensor array consists of several multiple beam antennas with a combined total number of beams of 256.

2. For efficiency, thinned beamspace processing is used. Specifically, the beams are grouped into 16 sets of beams with each set containing $K = 16$ beams.

3. The weight vector for each thinned beam set is calculated via sampled matrix inversion (SMI) or a computationally equivalent algorithm. Thus a 16×16 element covariance matrix $\hat{R}$ is calculated for each thinned beam set for each $2K = 32$ snapshots.

4. The complex sensor outputs x_i are sampled at the rate of 16 KHz to produce $256 \times 16,000 = 4 \times 10^6$ complex samples per second. Each sensor output is digitized into two eight bit word - one each for in-phase and quadrature data.

5. The product $x_i x_j^*$ required to evaluate the entries of the 16×16 covariance matrices are calculated in fixed point arithmetic at a rate of 4×10^6 complex multiplies per second. [For example, this function could be performed at a rate up to 20 MHz by the Plessey complex-multiplier, PDSP16112].

6. Subsequent operations are performed in floating point arithmetic.

7. Approximately $(K^2/2)2K = K^3 \approx 4000$ complex additions of the products $x_i x_j^*$ are performed at a rate of 4×10^6 floating operations per second (4 Mflops) to update the 16×16 covariance matrices $\hat{R}$. This is equivalent to 500 new covariance matrices per second or 32 covariance matrices per thinned beam set per second.

8. The $K^3 \approx 4000$ complex multiply-add operations (or 32,000 real operations) required to solve the Wiener-Hopf equation for each new weight vector (16 elements) are performed at a rate of 16 Mflops. Thus, 500 new weight vectors are generated per second, or equivalently 32 new weight vectors are generated for each thinned beam set.

Table 6.1 summarizes the floating-point computations required. A total of 20 Mflops are necessary. Thus, for example if commercially available array processor boards were employed, the required floating-point arithmetic could be performed by four boards dissipating total power of 60 watts (eg. Mercury Computer Systems MC3200 array processor board).

6.4.4 Calibration Subsystem

As indicated in Figure 6.1, a basic property of the proposed architecture is that the beamforming weights are applied in different physical paths from those used for making weight determination measurements. Path differences can result in degradation of the formed beam pattern, including loss of null depth, unless the differences are accounted for explicitly in computing the nulling weights. In addition, the desire to accommodate open-loop algo-

Computation Step	Flop (complex)	Flop (real)	Comp. Rate (Flops)	Update Rate/sec
Formulate an $\hat{R}$ matrix for $2K = 32$ snapshots	$\approx K^3 \approx 4000$ adds	8000	4×10^6	500
Solve Wiener-Hopf equation (or equivalent)	$K^3 \approx 4000$ mads	32,000	16×10^6	500

Table 6.1: Estimate of Real Floating Point Operations Per Second and Update Rate for Sizing Example

rithms also necessitates calibration of the hardware paths. These considerations lead to a need for on-board measurement and calibration hardware.

Figure 6.5 also indicates a calibration source whose purpose is to provide a single-tone reference signal for measuring relative path transfer functions and for calibrating the weight-setting attenuators and phase shifters. This signal is injected simultaneously into all input signal paths, between the LNAs and the weight-setting devices. The amplitude, phase and frequency of the calibration source is selectable under control of the nulling computer (computational and control subsystem). The LNAs should be switched "off" during system calibrations in order to avoid contamination of the calibration signal by possible received signal energy. The calibration process is assumed to be a background activity that is conducted periodically and/or during periods of communications inactivity.

Four types of calibration data are needed for the proposed system. These are:

1. Measurement Calibration Tables. These tables convert measured A/D counts in the $N + 1$ input channels into values of amplitude and phase shift, relative to a "reference" count for each device. These tables essentially linearize the input transducers. A separate table is needed for transducer in the system. Channel-to-channel relationships are established via inter-channel calibration constants. Calibration tables can be determined using on-board measurement resources.

2. Weight-Setting Attenuator and Phase Shifter Calibration Tables. These tables convert desired (analog) values of attenuation and phase shift into digital commands to be applied to the attenuator and phase shifter hardware. As in the case of the mea-

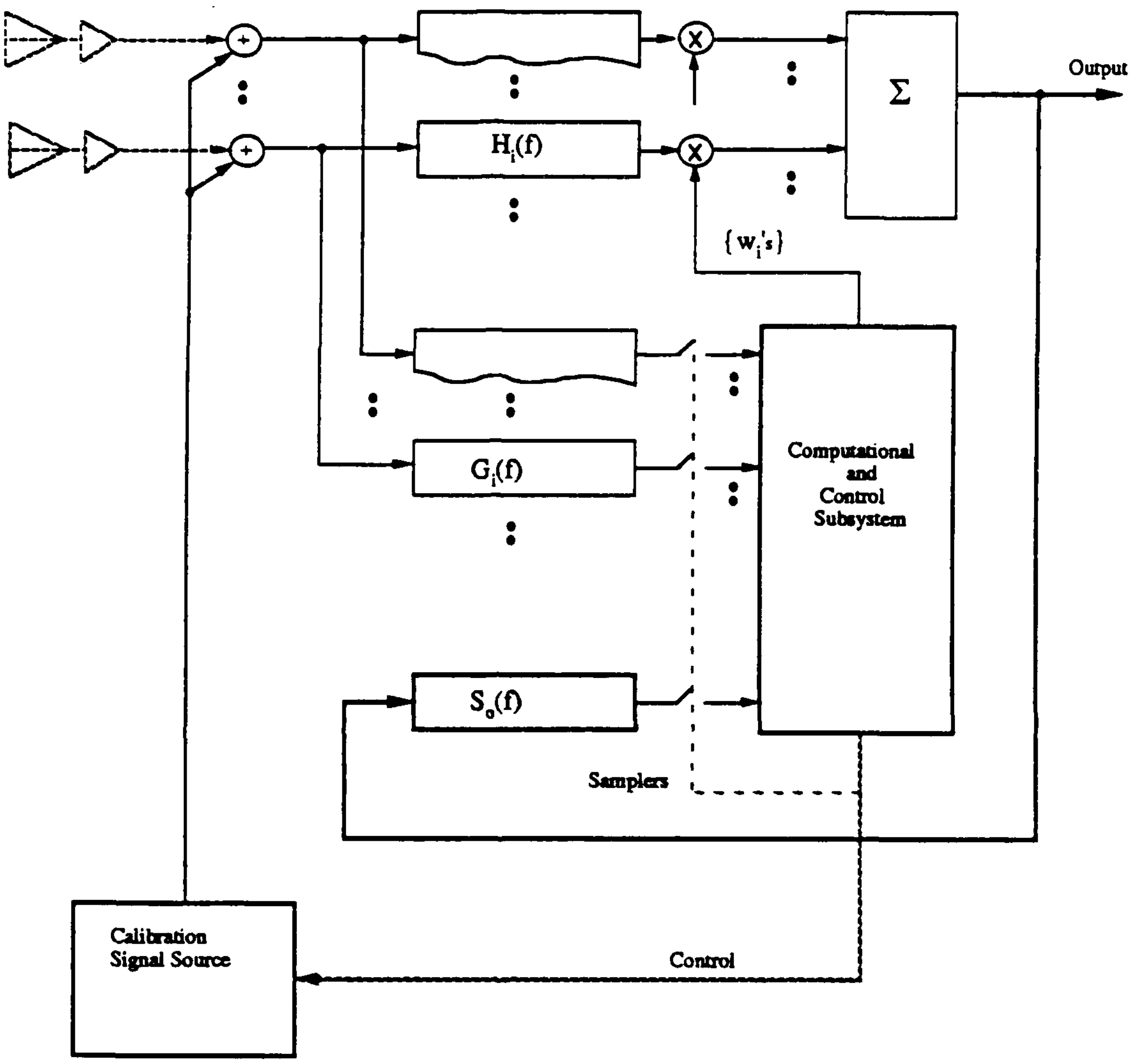


Figure 6.5: System Calibration Model

surement calibration tables, values are relative to arbitrary reference points in each device and channel-to-channel relationships are established separately via appropriate inter-channel calibration constants. A separate table is needed for each physical device in the system, i.e., an N element beamformer will require $2N$ calibration tables. These calibrations can be performed using on-board resources.

3. Inter-Channel Calibration Constants. These are divided into two types; i.e., measurement channel constants and weight-setting constants. Measurement calibration constants account for absolute level differences between the individual measurement channels, and possible gain variations between the antenna elements and the points at which the calibration source signal is injected. Weight-setting calibration constants account for channel-to-channel differences between attenuator and phase shifter reference settings, and possible channel-to-channel variations in the injected level of the local calibration signal. Inter-channel calibrations are best performed with the aid of a cooperative far-field source, i.e., a ground station.

4. Intra-Channel Calibration Constants. These account for transmission differences between the beamforming path and the measurement path in each receiver channel. The constants can be determined using on-board measurement resources.

In the proposed approach, calibration data is collected at a single frequency (e.g., band-center) using the calibration signal source shown in Figure 6.5. Alternatively, the same measurements can be made at several frequencies and averaged to achieve better accuracy. Representative single frequency measurement approaches are described below.

6.4.4.1 Measurement Calibration Tables

The procedure for deriving these tables requires that the calibration signal source be stepped through its entire range of signal amplitudes and phase shifts. An example procedure might be:

1. Eliminate received energy from outside sources. This can be accomplished by switching off the LNAs in all receiver channels.

2. Activate the internal calibration signal source. This causes the calibration tone to be injected into all channels simultaneously. Let the combined amplitude and phase setting of the source be Z_n, where n describes the commanded values.

3. Collect one or more snapshots of data via the complex samplers. A snapshot consists of $N+1$ samples, of which N (0 thru $N-1$) derive from the receiver channels and one (the N^{th}) is a sample of the beamformer output. The latter is ignored for this calibration run. Since the source signal is a sinusoid it will be possible to average a sequence of snapshots to achieve better measurement accuracy. Let the recorded data sample from the i^{th} channel be $x_i(n)$ and the average of a sequence of such samples be $X_i(n)$. Note that the calibration of this data requires the use of the measurement calibration tables described in Section 6.4.4.1.

4. Compute $X_i(n)/Z_n$ for $i = 0, 1 \ldots N-1$.

5. Repeat steps 3 and 4 for all signal source settings (all n).

The required calibration tables can be constructed from the above-collected data sets.

6.4.4.2 Weight-Setting Attenuator and Phase Shifter Calibration Tables

These calibration tables describe the relative transfer functions between all possible attenuator and phase shifter settings in a given channel. Data is collected in one receiver channel at a time by applying $w_i = 0$ in all channels except the one being measured. The attenuator and phase shifter in the channel under test are then sequenced through their entire ranges. At each setting one or more snapshots are collected, corrected via the measurement calibration tables described in Paragraph 6.4.4.1, and averaged. Let $Y_N(n_j)$ be the calibrated and averaged data collected in the beamformer output channel, where the argument denotes the joint setting of the digitally controlled attenuator and phase shifter in the channel under test. Also let the calibrated and averaged data in the j^{th} measurement channel be Y_j under these same conditions. Note that for $j \neq N$, Y_j is independent of the device settings since the latter are applied downstream from the measurement channel coupling junctions. Also, since all weights except the j^{th} are zero, the quantity $Y_N(n_j)$ contains energy only from the j^{th} channel. The processing consists of forming the ratios

$$Y_N(n_j)/Y_j, \text{all } j, \text{all } n_j. \tag{6.2}$$

The required tables are constructed directly from these computed quantities.

6.4.4.3 Inter-Channel Calibration Constants

The above-described calibration procedures concentrate on linearizing both the individual measurement channels and the weight setting hardware. The measurement transducers have thus far been characterized in terms of their linearity with respect to "reference" counts in each channel (Section 6.4.4.1). Channel-to-channel calibration constants are needed in order to relate these to a common reference level.

In the case of the weight-setting devices, it was assumed that for each value of i there exists a w_i whose value is considered to be equal to unity. This is the "reference" weight for the i^{th} channel. Let this be denoted as u_i. Channel-to-channel calibration constants are needed in order to relate the u_i to a common reference level.

The most direct method for establishing inter-channel relationships is by using an external cooperative source (ground station). The ground station is imagined to radiate a constant frequency pilot tone at the center of the communications band. Simultaneous measurement of the received signal level in all the measurement channels leads directly to the determination of the inter-channel measurement calibration constants. Similarly, the inter-channel weight setting constants can be inferred by shutting off (setting to 0) all but one beamforming weight at a time, and simultaneously measuring the output of the measurement channel in question and the beamformer output. The beamforming weight in the channel under test should be set to its reference value (u_i) for this measurement.

6.4.4.4 Intra-Channel Calibration Constants

Individual path transfer functions can be modelled as linear filters, as shown in Figure 6.5. The transfer function between the calibration signal injection point in the i^{th} channel and the beamformer output, is $H_i(f)w_i$, where f is frequency and w_i is the applied weight. Similarly, the transfer function between the calibration signal injection point in the i^{th} channel and the i^{th} channel measurement sampler, is $G_i(f)$. Finally, the transfer function between the beamformer output and the output measurement sampler is $S_o(f)$. Since the variation of these quantities is negligible over 2 GHz in the EHF communications band, the frequency variable can be ignored. Thus H_i, G_i and S_o are each modelled as complex constants over the band of interest.

Suppose a set of beamforming weights $\{W_i\}$ were computed on the basis of measurements alone. Since the data was viewed by the measurement system through the transfer functions

$\{G_i\}$, these gains are implicit in the results and the equivalent required weight set will be $\{W_iG_i\}$. Therefore, in order to achieve the desired beam pattern one would need to apply a modified set of weights $\{w_i\}$ in the beamforming path, where

$$H_iw_i = W_iG_i \tag{6.3}$$

This relationship results in a specification for the beamforming weights of the form

$$w_i = W_i(G_i/H_i) = W_iC_i \tag{6.4}$$

in which C_i is the i^{th} channel calibration constant. This constant is equal to the ratio of the complex (broadband) gains in the measurement and beamforming channels.

It is assumed that there exists a value $w_i = 0$ for which the transmission is zero. An example measurement procedure for determining the intra-channel calibration constants consists of:

1. Eliminate received energy from outside sources. This can be accomplished by switching off the LNAs in all receiver channels.

2. Activate the internal calibration signal source. This causes the calibration tone to be injected into all channels simultaneously.

3. Set $w_i = u_i, i = j$ and $w_i = 0$ all $i \neq j$

4. Collect one or more snapshots of data via the complex samplers. A snapshot consists of $N+1$ samples, of which N (0 thru $N-1$) derive from the receiver channels and one (the N^{th}) is a sample of the beamformer output. Since the source signal is a sinusoid it will be possible to average a sequence of snapshots to achieve better measurement accuracy. Let the calibrated data sample from the i^{th} channel be y_i and the average of a sequence of such samples be Y_i. Note that the calibration of this data requires Section 6.4.4.1.

5. Compute Y_j/Y_N. With the exception of the influence of S_o this is the required calibration constant, C_j, for the j^{th} channel. The presence of S_o can be ignored, as discussed below.

6. Repeat steps 3 through 5 for $j = 0, 1, ...N-1$.

A minor complication in the above calibration procedure stems from the fact that the data collected in the N^{th} sampling channel is a modified version of the beamformer output signal, i.e., it differs from the latter by the complex gain term S_o. The computed calibration constants C_i' will therefore differ from the desired quantities in accordance with

$$C_i' = C_i / S_o \tag{6.5}$$

The applied beamforming weights will similarly differ from their desired values by this same common factor. Introduction of a common factor into all the receiver channels does not affect the beamshape or the null depths or locations. The presence of S_o can therefore be ignored, at least as long as it is approximately constant over the entire communications band.

References

[1] L. Horowitz, H. Blatt, W. Brodsky, K. Senne, "Controlling Adaptive Antenna Arrays with the Sample Matrix Inversion Algorithm", *IEEE Trans. Aerosp. Electron. Syst.*, Vol. AES-15, No. 6, pp.840-848, Nov. 1979.

[2] L. Langston, S. Sanzgiri, K. Hinman, K. Keisner, D. Garcia, "Design Definition for a Digital Beamforming Processor", RADC Technical Report, RADC-TR-88-86, Apr. 1988.

[3] P. McKenzie, "In-Array Digital Beamforming Architecture", Atlantic Aerospace Project Report, *Army Contract DAAL02-87-C-0071* Feb. 1988.

[4] N. Owsley, "Systolic Array Adaptive Beamforming", NUSC Technical Report 7981, Sept. 1987.

[5] B. Potts, D. Cipolle, J. McHarg, A. Simmons, "An Experimental EHF Adaptive Nulling System", Lincoln Laboratory Technical Report 606, Sept. 1983.